UTILIZING BAYESIAN TECHNIQUES FOR USER INTERFACE INTELLIGENCE

THESIS

Robert Allen Harrington
First Lieutenant

AFIT/GCS/ENG/96D-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/96D-08

UTILIZING BAYESIAN TECHNIQUES FOR USER INTERFACE INTELLIGENCE

THESIS

Robert Allen Harrington
First Lieutenant

AFIT/GCS/ENG/96D-08

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GCS/ENG/96D-08

UTILIZING BAYESIAN TECHNIQUES FOR USER INTERFACE INTELLIGENCE

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Robert Allen Harrington, B.S.
First Lieutenant

December, 1996

Acknowledgements

This research has been the most important part of my experience at the Air Force Institute of Technology. I began with the common question "What is research", and finished with a sincere understanding of the importance research holds for the United States Air Force. I'm sure this experience will prove invaluable throughout the remainder of my Air Force career.

I wish to extend a hearty thanks to Dr. Eugene Santos Jr. and Maj Sheila Banks for their thoughtful questions and relentless press for success. Their encouragement and wisdom was paramount to my research success. I also want to thank Dr. Thomas Hartrum for his time in reviewing my research documents and his inspiring teachings in software engineering. Yes Virginia, software engineering is an engineering discipline.

A special thanks goes out to my fellow members of the PESKI research team: Brett, Louise, Dan, Ed, and Scott. Their teamwork was the key in completing the first version of PESKI. Also, I shouldn't forget to thank Chip, my trusty steed of the information super-highway.

Finally, I'd like to extend a heartfelt thanks to my wife, Sherry, and my son, David, for their support and patience. I recognize the hardships they have endured, dealing with an overly busy husband and father. I couldn't have done it without you both!

Robert Allen Harrington

Table of Contents

List of Figures

Abstract

Software systems are becoming increasingly complex, so complex that ordinary users are often overwhelmed by windows, buttons, menus, and a multitude of system functions. In order to keep software systems usable, we must begin to abstract the complexity of these systems away from the user, in other words, make the complexity of software systems transparent to the user. This abstraction must occur within a system's human-computer interaction, so the responsibility for providing this abstraction falls on the software system's user interface.

The purpose of this research is to study the injection of an intelligent agent into modern user interface technology. This agent is intended to manage the complex interactions between the software system and the user, thus making the complexities transparent to the user. The background study will show that while interesting and promising research exists in the domain of intelligent interface agents, very little research has been published that indicates true success in representing the uncertainty involved in predicting user intent.

The interface agent architecture presented in this thesis will offer one solution for solving the problem using a newly developed Bayesian-based agent called the Intelligent Interface Agent (IIA). The proof of concept of this architecture has been implemented in an actual expert system, and this thesis presents the results of the implementation. The conclusions of this thesis will show the viability of this new agent architecture, as well as promising future research in examination of cognitive models, development of an intelligent interface agent interaction language, expansion of meta-level interface learning, and refinement of the PESKI user interface.

UTILIZING BAYESIAN TECHNIQUES FOR USER INTERFACE INTELLIGENCE

I. Introduction

1.1 Problem Description

Software systems are becoming increasingly complex, much to the chagrin of system users. System developers and researchers have turned to improving the human-computer interaction of software systems to reduce the complexity faced by users and make systems more usable. One of the latest areas of study into improved human-computer interaction is the study of integrating intelligent agents into modern user interface technology. It has been hypothesized that this intelligence can be used to provide complexity abstraction and interface adaptation in order to make the user interface, and thus the software system, more usable to the system user. However, researchers have failed to find a proven architecture for interface intelligence agents that can be widely adopted to today's user interface technology.

This problem is observed more acutely in the arena of expert systems. Traditionally, designers of expert systems (computer systems that imitate human reasoning and knowledge) have placed little or no emphasis on the expert system user interface. These developers spend most resources perfecting the functionality of the expert system, leaving few resources available to design and implement the user interface and enhance the user's ability to utilize the system. Therefore, expert systems of the past are known for their inflexible, non-user friendly formats that are designed around the expert system's domain rather that the user's needs. This has begun to change as expert systems become more widely used among common computing systems, forcing interface design issues into the development spotlight.

The need for more dynamic, user-oriented expert system interfaces becomes even more crucial with the development of generic expert systems, systems that can be used in and adapt to many domains, such as EDWARD [19], OPADE [23], and PESKI [75]. This type of expert system is available to a wide variety of users with a wide variety of preferences and needs. The user interface for a generic expert system must be the catalyst for handling a variety of user types and must employ a level of intelligence to manage the users' preferences and needs. The creation of an effective, intelligent user interface can support an expert system's performance throughout a myriad of application domains and is crucial to user acceptance of the system as a useful, cost-effective tool.

Exploration of interface intelligence is vital to United States technological goals as well. The High Performance Computing, Communications, and Information Technology Subcommittee (HPCCIT), tasked by the President with advancing the United States technological advantage, places emphasis on user interface research as the catalyst for handling complex human-computer interaction for many of the National Challenge applications [5, 3, 6, 2]. In fact, the HPCC FY 1994 Blue Book directly names intelligent user interfaces as one of a three-part technology base for meeting National Challenges [1].

1.2 Research Goals

The four goals for this research address the problem description discussed in Section 1.1. All work performed for this research is oriented toward meeting these goals:

1. To develop an interface intelligence agent architecture that accurately captures and models user intent.
2. To develop an interface intelligence agent architecture that abstracts the complexity of an expert system away from the user.
3. To develop an intelligent user interface that may be applied across many domains while maintaining the generic nature of a generic expert system.
4. To show the viability and usefulness of Bayesian techniques for interface intelligence.

1.3 Scope

The study of user interface intelligence spans a wide variety of disciplines, as will be presented in Chapter II. Therefore, the scope of this thesis is to study the application of Bayesian theories and practices to develop a new, more effective agent architecture called the Intelligent Interface Agent (IIA) [21]. The proof of concept of this architecture is executed in a designed and implemented user interface supporting the generic expert system PESKI (Probabilities, Expert Systems, Knowledge, and Inference) [75, 13, 35, 36, 37]. The IIA uses a probabilistic network that stores user behavior collected from the interface's human-computer interaction. The IIA also employs a reasoning mechanism that uses the stored probabilities to make decisions on future user intent. Other user interface architectures and issues involving cognitive science concerns are presented in Chapter II and Chapter VIII, but they are not the primary focus of this research.

1.4 Thesis Overview

Chapter II of this thesis discusses the wide variety of research areas that make up the study of user interface intelligence. Chapter III explains the methodology used to carry out this intelligent interface agent research. Chapter IV explains the design process used to develop interface intelligence. Chapter V outlines how the

designs were implemented, pointing out important implementation decisions and tradeoffs. Chapter VI of this thesis features the test cases and test results of the newly implemented IIA. Chapter VII of the thesis presents an interpretation of the test results and the conclusions posed by this research. Finally, Chapter VIII describes several interesting and promising future research areas that are spawned from the conclusions of this research.

II. Background

The study of intelligent user interfaces is the focus of knowledge from many varied areas of science, having roots in computer science, psychology, and human factors engineering [76, 12]. Human-computer interaction (HCI) research, which combines knowledge from all three areas, supports the study of adaptable and adaptive user interfaces. The merger of adaptive user interface research and artificial intelligence knowledge representations, such as Bayesian networks, sets the stage for intelligent user interface research. This chapter traces the history of interface intelligence and provides a foundation that supports the development of this research.

2.1 Human-Computer Interaction (HCI)

The last fifteen years of computer development have brought about a focus on HCI research [29]. Numerous journal publications, conferences, and books are dedicated to the advancement of HCI research, all searching for better ways for computers to interact with humans. In order to understand HCI better, we must first understand the user and the interface.

2.1.1 The User. A good researcher must understand users in order to study user interfaces. Cox explains there are many characteristics to the typical user including frequency of use, application knowledge, tasks to be performed, assumed skills, and attitudes [29]. There are also many human factors relating to computers that can be measured. These include time to learn the system, speed of task performance, rate of errors, retention of system knowledge over time, and satisfaction [76].

Researchers continually attempt to categorize users into general user classes [84, 76, 45, 86]. Trumbly asserts that user performance is improved when the interface characteristics match the user skill level. Some user categories include novice,

knowledgeable intermittent user, and expert. Novices usually know little about their tasks and have poor computer skills. They tend to prefer menus that require very little of the user's short term memory to use. Knowledgeable intermittent users usually know something about the tasks and have some computer skills. This type of user may prefer either menus, keyboard commands, or a mixture of both. Experts are very familiar with their tasks and have a high degree of computer skills. These experts generally prefer keyboard commands so that they may perform tasks more quickly.

2.1.2 The User Interface. One of the most important components of today's modern computing systems is the user interface. The user interface is the communication link between the system user and the functionality of the system [62]. A user interface does not have to be graphical in nature, although many modern interfaces rely heavily on graphical representation [76, 29]. Donskoy defines the user interface as a set of objects that are affected by user and system events [30]. Since all of the user's interaction with the computing system is with the user interface, the interface plays a vital role on the user's opinion of the system as a whole [34, 89].

Cox suggests there are four main characteristics of a good user interface: user control, transparency, flexibility, and learnability [29]. He goes on to state that some general design principles that lead to a good user interface are consistency, keeping the user in control, easy reversal of adaptations, and user modifiability. Schniederman suggest dialogue between humans and computers must be consistent, allow for shortcuts, allow feedback, and ensure the user knows when a dialogue is done [76].

Unfortunately, the user interfaces of today's systems have become increasingly complex. Woods suggests that the data overload humans exhibit introduces new kinds of errors into system use, errors that are born out of having too many choices

and too much data [89]. Ways must be found to reduce the complexity of user interfaces and avoid these errors.

There are many different ways a user interface can communicate with the user [45]. These methods fall into three general categories: language, graphical, and multimodal [19, 76].

2.1.2.1 Language Communication. Language communication includes textual and natural language conveyance of information and deals effectively with abstract ideas and concepts [39, 9, 77]. Textual language communication is a simple, static method in which the interface can respond to specific textual inputs. Natural language broadens the interface's ability to understand textual input by allowing the input to be typed in a real language, such as English. However, current language systems are very restricted, offering only limited capability, and may require lots of typing to express ideas. Further, natural language technology faces numerous barriers to offering practical natural language capabilities.

2.1.2.2 Graphical Communication. Graphical communication involves the viewing and manipulation of physical objects and is ideal for concrete tasks and procedures. One of graphics downfalls is interpretation, since graphical symbols can mean different things to different people. Also, implementation can be a problem for graphics since they are very resource intensive [62, 76].

The graphical user interface (GUI) utilizes graphical communication to convey information to the user in abstract ways for easier comprehension [14, 25, 88]. The most important element of making this type of user interface effective is providing a representation that has the proper semantics and syntax. Unfortunately, these graphical representations can often have the downfall of being inflexible, since the representations are statically defined when the interface is first designed and implemented.

Windowing systems are currently the most popular way for systems to represent information graphically. Among these windowing system, the X Window System, normally called X, stands out as a popular choice, mainly due to its portability to systems such as UNIX, MS-DOS, Windows NT, SUN/XView, and OpenVMS [88, 50]. Bernstein describes the X as "a protocol that is exchanged between two processes; a client(typically an end user application) and an X server that drives the keyboard/mouse devises and renders images on the screen" [17]. As a standard protocol, window toolkits use X to create an environment of graphical objects, called widgets, such as windows, buttons, and icons. There are many available toolkits that can be applied to the X Window System, such as Athena, OLIT, OPEN LOOK, DECwindows, and OSF/Motif, and each toolkit has its own look and feel.

The Motif toolkit has emerged as a very popular set of widgets to implement a GUI. Motif is made up of widgets written in the C and C++ programming languages, and the Open Software Foundation specifies the look and feel of Motif to ensure the toolkit is maintained as a standard. Motif is the most popular toolkit because of its adherence to modern object-oriented standards of software development and its ease of implementation [88, 72, 55].

Graphical elements, such as icons, provide a way to abstract representations for improved communication with the system user [14]. Icons must be designed and used very carefully since they can be misinterpreted. Use of icons can lead to creation of direct manipulation systems [27]. These systems allow the user to manipulate icon images, mimicking the actual manipulation of the real world. For example, a car designer can match hubcaps to a newly designed car by moving different hubcap icons onto a graphical representation of the car. Graphical manipulation systems can be very easy for users to learn and use.

2.1.2.3 Multimodal Communication. When an interface has communication methods that come from both areas, language and graphical, and it offers

a user the choice between its methods of communication, it is known as a multimodal interface [19, 65, 27, 77]. Such interfaces as AlFresco [78, 79] provide rich interaction environments for the user. In fact, Neilson suggests that a combination of language and graphical communication may actually be more effective in concert than separate [62].

2.2 Adaptable and Adaptive User Interfaces

Research concerning intelligent user interfaces began in the late 1980's with the examination of adaptable user interfaces. Adaptable user interfaces include multimodal interfaces and interfaces that allow users to modify specific elements of the interface's operations and appearance [48, 67]. Systems that support user controlled multimodalty such as GUIDE [48], EDWARD [19], and AlFresco [78, 79] allow great flexibility to the user while offering access to the software system's complex functions. For example, AlFresco (an art evaluation tool) allows the user the choice to communicate with the software system through menus or through on-screen, mouse pointer cues using multimedia pictures of real art objects. The adaptable nature of this interface allows AlFresco users to control the interface to meet their personal needs and desires.

Unfortunately, many user interface adaptations are very static in nature, being designed by the interface designer when the interface is created [15, 51]. These adaptations may also require a certain level of expertise or interactive help just to enact. Due to these restrictions, the resulting adaptations are limited in scope and usefulness to the user [48, 19, 78, 86].

Interface adaptation research led to the development of adaptive user interfaces that adapt themselves based on perceived user behavior [82, 15, 51, 45]. Maulsby describes user behavior as "actions that might be performed in one situation but not another" [58]. This act of behavior prediction is also known as plan recognition [87, 46, 10].

Trumbly states that "Adaptive systems can't adapt to every user perfectly but can adapt enough to improve the quality of the human-computer interaction" [84]. Trumbly goes on to point out Valery Venda's law of mutual adaptation, where "the performance of a task is best when the capabilities of the computer match the cognitive skill structures and behavior strategies of the human user" [84]. These systems must collect data about the application, the system user, and the application domain in order to have enough information to make intelligent decisions. Opperman suggests that systems must be able to adapt themselves to the user based on metrics such as errors, characteristics, performance, and goals [67].

The decisions made by these systems must be collaborative [82, 15, 11, 51, 16, 61]. The adaptive interface proposes the adaptation, presents evidence to support the need for the adaptation, and the user accepts or rejects the suggestion [67, 60, 66, 49]. Ruckert asserts humans should be able to discuss results with a knowledge-based decision support system [71]. However, this collaboration must never detract the user from the actual application at hand and must, in the end, provide a cost benefit to the user [82, 19, 81]. Miller wrote about this interaction from the computer to the user, saying "Justification builds trust" [60]. Once the adaptation is complete, the system should give the user an overview of the adaptation and the ability for the user to reverse it [67, 82, 85].

There are many behaviors that can be exhibited by an adaptive user interface [45, 16]. They include reorganizing menus and dialogue boxes, streamlining task procedures, enabling and disabling interface widgets, and providing help [80, 44, 51]. Adaptive interfaces, such as Flexcel [67], Retail User Assistant [59], and UIMS [44] collect metrics about the user while the user employs the system and then applies those metrics to make decisions about desirable changes to itself.

Flexcel stores knowledge about the user in a series of If-Then rules. Flexcel tracks error rates, adaptation preferences, and thresholds for initiating rules. For example, when a threshold is met for too many errors while doing a particular task,

a dialogue is presented to the user suggesting how that user can prevent the same error in the future. If an adaptation is presented to solve the problem, the user can indicate acceptance of the adaptation and Flexcel will perform it [67]. Retail User Assistant (RUA) also collects data about the user and adapts itself based on the learned knowledge being applied to a rule-base. RUA is a multimedia package used in computerized cash registers that provides video directions on how to perform register operations. The system starts by giving basic instructions to new employees and, as a particular employee gets better at operating the register, automatically eliminates cues on operations that the employee has mastered [59]. UIMS is a multimodal interface manager that alters its modes of communication based on a dynamic set of metrics collected from each systems user [44].

It is very difficult to measure the success of an adaptation. Kuhme writes, "There is yet no generalized metric for a systematic, system-built-in evaluation of a performed adaptation" [51]. He even goes on to suggest that most adaptive user interfaces do not meet the needs of the user. In general, adaptive interfaces are observed to be helpful and productive in numerous application areas, but many users are still uncomfortable with the idea of automated computer system control [67]. Woods suggests that the user can think of a totally autonomous agent as a type of crutch, treating the user like they don't know what they're doing [89]. Unfortunately, many adaptive interfaces, such as ChEM [11], are also very inflexible since they are only designed for specific applications.

2.3 The Intelligent User Interface

The intelligent user interface is an attempt to overcome some of the shortfalls inherent in adaptive interface technology by adopting a more sophisticated intelligence. Woods describes an intelligent interface as an "autonomous, animate computer agent" or "smart instrument" and states that it can also be characterized as an agent that can help the user deal with state changes of the application [89]. The

idea of developing a cooperative user interface, or intelligent assistant [57], represents a compromise between improving user interface management of complex tasks and allowing users to stay in control [24, 12, 61]. An intelligent assistant will collect metrics about the system's use and look for areas where adaptations can occur. When target adaptations are identified, the interface notifies the user and asks for permission to make the target adaptation. It is important to note that this method of consulting with the user before making an adaptation must not distract the user from the user's task.

A sophisticated intelligent assistant assists the user with performing tasks by allowing the user to ask for tasks in an abstract way, leaving the assistant to perform the details [49, 24]. Chappel wrote, "Many intelligent interfaces include knowledge of the application task, user and context for pragmatic planning and natural language use." [25]. Woods claims designers of intelligent interfaces should tread cautiously to avoid creating new examples of clumsy automation [89].

Puerta suggests there are three major functions that are performed by an intelligent user interface: knowledge-based interaction, self-adaptation, and automatic generation [69]. Knowledge-based interaction deals specifically with performing more intelligent communication with the user. Self adaptation involves the intelligent interface improving its own performance based on perceived and stored behavior. Automatic generation takes place when the interface uses its existing knowledge base to generate an interface that is particular to the current state of the knowledge base.

Chappel describes a system called MMI2 (Multi-Modal Interface for Man Machine Interaction with a Knowledge Base) [25]. This interface uses statistical knowledge about users, applications, tasks, and dialogue context to create natural language dialogues and graphical responses for a computer network design tool. This system can use its knowledge to chose an appropriate graph to represent information that the user wants to display, such as displaying the cost of all computer hardware in a newly designed computer network. MMI2 uses a static knowledge base and Chap-

pel concludes, "Clearly the ideal would be to incorporate perceptual and cognitive models that would be used to simulate the user and show the understanding gained from a presentation and any false inferences and implicates drawn" [25].

Current intelligent user interface research has found new energy with the study of intelligent agents. An intelligent agent is characterized as a computer entity that collaborates with and helps a user [12, 61]. Hayes-Roth describes the roles of an intelligent agent as "perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions" [42]. These roles match the previously discussed roles of today's intelligent user interfaces. Intelligent agent research is useful to intelligent user interface research because it supports creating compact, portable intelligence that is independent of a particular application interface. This idea of a generic interface intelligence pushes the limits of intelligent interface research.

2.4 Knowledge Representation

Intelligent user interface research is very focused on human-computer interface issues, concerned mainly with the abilities and usability of intelligent interfaces. However, intelligent interface research has put little emphasis on improving the structures that represent the intelligence of these interfaces. Most research interfaces use static, rule-based intelligence structures rather than exploring more dynamic representations [34, 56, 25, 82].

It is widely agreed that basing decisions on accurate cognitive models of the user is important for accurate prediction of user intent [25, 82, 67, 84, 89]. It has also been supported that the interface should be able to collect and model information about false inferences. Many agree collecting such data is extremely difficult for reasons of computational efficiency and cognitive modeling [44, 51].

The problem is that most intelligent user interfaces that use rule-based representations fail to correctly represent uncertainty [67, 59, 44]. Therefore, knowledge representations that can capture and model uncertainty in human-computer interaction can improve the modeling of the user and the user interface's behavior [44, 51]. One knowledge representation that is ideal for representing uncertainty is a Bayesian Network(BN).

A BN is a mathematically correct model for representing uncertainty that shows probabilistic relationships between items [8, 43, 26, 68]. Formally, a BN is a directed acyclic graph where each node represents a random variable of interest, and edges represent direct correlations between the variables [20]. Burnell writes, "Belief networks, also known as Bayesian belief networks or probabilistic influence diagrams, are a flexible and powerful approach to representing uncertain knowledge" [22].

The topology of a BN consists of nodes and arcs [26]. The nodes of a BN represent random variables, or states of affairs. The arcs of a BN represent relationships between the nodes [63]. Since a BN is a directed graph, the nodes are unidirectional and the direction represents dependency. A node that has no dependencies is known as a root node, and a root node contains probabilities for the states of the node, usually true and false. A node that is not a root node contains conditional probabilities that are derived from all possible combinations of its descendent nodes.

The conditional probabilities of non-root nodes represent a degree of belief [43, 26]. Beliefs can be represented by probabilities, defining the likelihood an event will occur. A probability of 1.0 indicates certainty that the belief is certainly true, and a probability of 0.0 indicates that the belief is certainly false. Any probability between 0.0 and 1.0 represents a degree of uncertainty that the belief is true or false [52, 9]. Because of this fact, probability theory is considered one of the strongest ways to represent uncertainty [63].

Inferencing in a BN is the act of deriving probabilities from the data in the network [68]. There are two main types of inferencing associated with BNs: belief revision and belief updating. Belief revision derives probabilities of all the nodes in the network, often referred to as *the state of the world.* Belief updating derives the probabilities of one particular node of the network.

BNs also allow for an easy way to designate conditional probabilities, normally represented by conditional probability tables (CPTs) [20]. While these CPTS are exponential in nature, they provide a basic way codify conditional probabilities [31]. Unfortunately, the exponential nature of CPTs are a significant problem when implemented computationally. In fact, computing a BN is NP-hard, and this computational problem is the largest barrier against widespread use of BNs [26].

BNs have the property that they support reasoning for many types of problems including medical diagnosis, map learning, language understanding, and vision [26, 22, 8]. Nicholson writes about the usefulness of BNs for collecting events from real world sensors, such as a light sensor, in order to create a probabilistic model of the real world. This model can be used in areas such as robotics to aid in machine decisions about the real world [64].

Bayesian techniques offer attractive properties for developing interface intelligence. They provide an excellent ability to capture uncertainty, something inherent in modeling human intent. Also, Bayesian techniques are extremely useful for predicting future events. Finally, these techniques are useful in expressing qualitative relationships (causalities) among beliefs and to process these relationships in a way that yields intuitively plausible conclusions [68].

III. Methodology

Development of a user interface agent for this thesis follows a six phase process: adopt a target system, develop interface requirements, design the interface, design the intelligent agent, implement and integrate the designs, and test the intelligent agent's effects on the target system [55, 29, 76].

3.1 Adopt a Target System

The first step in developing interface intelligence for this research is to identify a system that needs intelligence in its interface. Systems that need interface intelligence might include very complex systems, systems that require a very dynamic collaboration with the user, and systems that must adapt to varying application domains. Further, it is desirable to target a system that already has an acceptable, working user interface. This avoids the effort necessary to design and implement the user interface and allows the researcher to focus only on the intelligent agent. It can be argued that intelligent interface research can be accomplished with a simulated target system rather than a real system. However, development of interface intelligence for a real system lends credence to the research by exposing real world performance and usability issues of the interface

3.2 Develop Interface Requirements

Once a target system is identified, the system design, the application domain, and the potential domain users must be analyzed to develop interface requirements. These requirements should focus on user needs and be tempered by target system capabilities [55]. A comparison of tradeoffs may be necessary to find the right balance. Intelligent interface requirements are then decomposed into specific requirements for both the physical user interface and the intelligent agent. This decomposition facilitate the mapping of requirements during the design phases.

3.3 Design the Interface

Once requirements for the physical interface are derived, those requirements are transformed into an interface design. This design begins with storyboarding, a visual interface rapid prototyping technique that shows what the interface will look like and how it will react to events [29, 53, 54]. Approved storyboards are transformed into object-oriented models [72, 40]. These models provide abstract representations of the interface objects and events for easier implementation into operational code.

3.4 Design the Intelligent Interface Agent

The requirements for the intelligent agent, along with the design of the physical user interface, are used to design an intelligent interface agent. This agent architecture is developed to both mathematically and semantically meet the intelligence requirements. This architecture must also allow efficient communication with the physical interface and meet usability standards. The design of the candidate intelligence structure can be characterized in four steps: develop the semantics, design the knowledge representation, develop a learning method, and develop an inferencing method.

3.4.1 Develop the Semantics. First, the semantics are developed based on intelligence requirements. These requirements are used to discover interactions between the user and the application domain that requires intelligent intervention. The semantics should capture relationships between the relevant elements and describe how the elements interact.

3.4.2 Design the Knowledge Representation. Second, a knowledge representation topology is designed that supports the semantics. This representation should clearly mirror the relationships and interactions of the relevant elements. Further, this representation should be implementable and computationally efficient.

3.4.3 Develop a Learning Method. Third, a learning method is designed. This method should ensure totally transparent elicitation of knowledge from the user and the application domain. The learning should correctly transfer knowledge from the domain, translate the knowledge into the representation, and store the knowledge efficiently. The computational efficiency of this processes is a key element to ensuring learning transparency.

3.4.4 Develop an Inferencing Method. Fourth, an inferencing method is designed to support knowledge elicitation from the representation. This inferencing method must be able to access the relevant knowledge in a computationally efficient manner, on demand, and must retrieve the knowledge in a way that ensures the returned knowledge is correct.

3.5 Implement and Integrate the Designs

The final designs of the physical interface and the intelligent agent are then implemented into programming code. The choice of computing platform and language should be based on requirements such as the target system hardware and software environments. This implementation is a stepwise process of software implementation and software testing [47]. Coding accuracy is checked by developing and running tests that exercise all aspects of the implemented software.

Once the physical interface and the intelligent agent are implemented and tested, a process of system integration begins. This process starts by integrating the interface and the agent into an intelligent user interface. Once software testing is performed on the intelligent user interface, the interface is integrated with the target system.

3.6 Test the Integrated Intelligent User Interface

Finally, the interface intelligence is tested to determine its ability to meet the requirements [83]. Test case design focuses on exercising the agent's mathematical accuracy and usability. Enough test cases must be performed to examine all areas of performance. Testing for mathematical correctness is accomplished by comparing the results of hand-computed calculations with actual intelligent interface results for a particular test case. These test cases are designed using a foundation of accepted mathematical formulas and constructs. Usability testing is performed with standard time/step analysis and user feedback sessions. Together, the results of these tests will give an indication whether the research goals are met.

IV. Design of the Intelligent User Interface

4.1 Adopting the Target System

As mentioned in Chapter 1, PESKI is chosen as the target system to develop the interface intelligence. PESKI is a collection of expert system tools that form a generic expert system, able to provide reasoning and decision support capabilities independent of any application domain [75, 13]. At the start of this research, PESKI did not have a user interface nor a system architecture that binds the expert system tools. An incomplete system provides some difficulties in developing interface intelligence because of the potential resources needed to create an interface and complete the system architecture. However, this generic expert system possesses the complexity and requirements for dynamic human-computer collaboration ideal for incorporation of interface intelligence.

4.2 Developing Interface Requirements

Requirements for the PESKI user interface include inputs from interviews with the lead PESKI researcher, an examination of expert system tools to be integrated under PESKI, a study of possible PESKI users, and a study of communication modes that meet the needs of the targeted users.

4.2.1 General Interface Requirements. Interviews with the lead PESKI researcher uncovered three main requirements for the PESKI user interface. First, the interface must be generic, not tied to any particular application domain. Second, the PESKI interface must be flexible to all manners of user preferences and abilities since it is a system that is intended for all types of users, from novices to experts. Since PESKI does not have a system architecture, the third main requirement is to provide the architecture. The ideal situation is to design and implement a system architecture separate from the interface, but applying research resources to the de-

velopment of a separate architecture was unrealistic. Therefore, the user interface is also required to be the PESKI system architecture.

4.2.2 Examination of PESKI Tools. As mentioned earlier in this thesis, PESKI is a generic expert system, designed to provide users from a multitude of application domains with decision support capabilities. To do this, PESKI offers various independent tools that, when combined under one system, provide the required functionality. All these tools share one thing in common: they all utilize and support a new knowledge representation called a Bayesian Knowledge Base (BKB) [75, 13].

There are six basic tools that support PESKI's knowledge representation [18, 33, 74]. The knowledge acquisition (KA) tool is primarily responsible for putting knowledge into the BKB. The edit supports (ES) tool supplements the knowledge acquisition tool. The inference tool (IE) is responsible for extracting knowledge from the BKB. The verification and validation (VV) tool is responsible for discovering incomplete or contradictory information in the BKB. The data mining (DM) tool is used to extract information from outside sources for use in the BKB. Finally, the knowledge base viewing (KV) tool is necessary for realizing more abstract ways of viewing BKBs.

4.2.3 Study of Possible PESKI Users. The study of possible PESKI users is important to the success of the entire system. After all, a computer system is useless if the user does not perceive the system as a convenient tool to get useful work done. Furthermore, if the system's interface is developed without a focus on user needs, the system may prove to be difficult to operate and be promptly labeled worthless.

It is sometimes difficult to develop a system that consistently and accurately predicts user needs in a generic environment. For example, new user needs are extremely difficult to predict since the interface has no past behavior on which to

base predictions. Therefore, we must attempt to classify general groups of users and develop user profiles for each group. Although it is very difficult to put users into specific classifications [73], doing so will narrow down the requirements of the interface to a manageable level. With this in mind, a general purpose expert system, such as PESKI, is assumed to have four general types of users: *application users*, *application experts*, *knowledge engineers*, and *computer scientists*.

The *application user* employs the expert system "on-the-job." This user is not necessarily an expert in the application domain nor is this user necessarily knowledgeable in the inner workings of an expert system. This user simply needs the ability to query the expert system for the appropriate knowledge to get useful work done.

An *application expert* is extremely knowledgeable in the field where the application is being used. For example, an application expert is a doctor in a hospital. This user is not necessarily knowledgeable in expert systems, but may wish to use the system to query for some required knowledge. Since application experts are experts in their domain, these users may also be responsible for providing new knowledge to the existing system.

A *knowledge engineer* is a person who specializes in acquiring knowledge, usually from humans. This user acts as an intermediary between experts and the computer, able to break knowledge down into its most basic forms for entry into the expert system. This user needs a more detailed view of the knowledge base than the application expert or the application user. The knowledge engineer also needs to view and analyze the structure of the knowledge in the knowledge base to ensure that the knowledge is being built correctly.

The *computer scientist* is also an important user to consider for an expert system interface. This user is interested in development, design, and maintenance of the expert system and needs the lowest or no amount of abstraction. The user interface must allow the computer scientist to directly manipulate knowledge base structures,

to examine knowledge interactions, and discover how to improve knowledge structuring techniques.

The users of PESKI may fall within more than one of these general categories. For example, a civil engineer can use the expert system to design a building's foundation but can also add new information concerning building design to the knowledge base. Therefore, the civil engineer is an application user and an application expert. The interface must allow users who maintain multiple roles the ability to switch between the needs of the various roles while using the system.

Not only are there general classifications of PESKI users but there are various skill levels in computer knowledge among individuals of various groups. A widely accepted scale of computer knowledge is low, medium, and high [84]. This computer knowledge can range from general knowledge to specific knowledge in particular applications. For example, a user that is very knowledgeable about particular word processing systems may have no idea how operating systems work.

4.2.4 Study of PESKI Communication Modes. Three communication modes are proposed for this interface to meet the needs of the previously mentioned users: *natural language*, *graphical manipulation*, and *structured text*. The user interface should provide easy access to all three communication forms and should assist the user with choosing the appropriate communication form for each application or tool [19].

The *natural language interpreter(NLI)* allows the user to communicate with the expert system through typed sentences of English text [39, 9]. For example, a doctor can create a relationship between "polio" and "disease" in the knowledge base by typing "Polio is a disease" onto a text line. The interpreter breaks the sentence into its most basic forms, translates the forms into a relationship, and sends the update request to PESKI's knowledge acquisition tool.

The *graphical interpreter* allows the user to build graphical relationships that can later be manipulated to communicate relations [11]. For example, an icon that represents "polio" can be moved, using a mouse, from a pool of icons to a window that represents "diseases." The graphical communicator interprets the move as creating a relationship between polio and disease and sends the update request to PESKI's knowledge acquisition tool.

The *structured text interpreter* allows the user to build pre-defined textual relations that, when receiving text, create the desired relation. Again, if "polio" needs to be associated with "disease," then a text entry box is created labeled "disease." The user can then type "polio" in the box and the structured text interpreter creates the relationship and sends the update request to PESKI's knowledge acquisition tool.

4.3 Designing the User Interface

The interface architecture developed from the requirements is a layered architecture that contains three main layers: the graphical layer, the system layer, and the IIA layer(see Figure 4.1) [38]. The graphical layer provides the graphical interface environment, or cosmetics, for the communication with the user, and is responsible for managing the graphical user interface. Due to the event driven nature of the modern graphical user interface and the lack of a PESKI system architecture, the graphical layer is also deemed responsible for overall system control. The system layer provides a coupling between the expert system's tools and the user interface. The IIA layer manages the interface intelligence, communicates with the graphical user interface, communicates with the system layer, and manages the communication modes.

Each of these three main layers of the interface architecture are represented by object classes. The IIA layer is discussed later in this thesis. The system layer is designed to support all the main expert system tools except knowledge viewing. Driver classes are designed in the system layer to support each of these functionalities

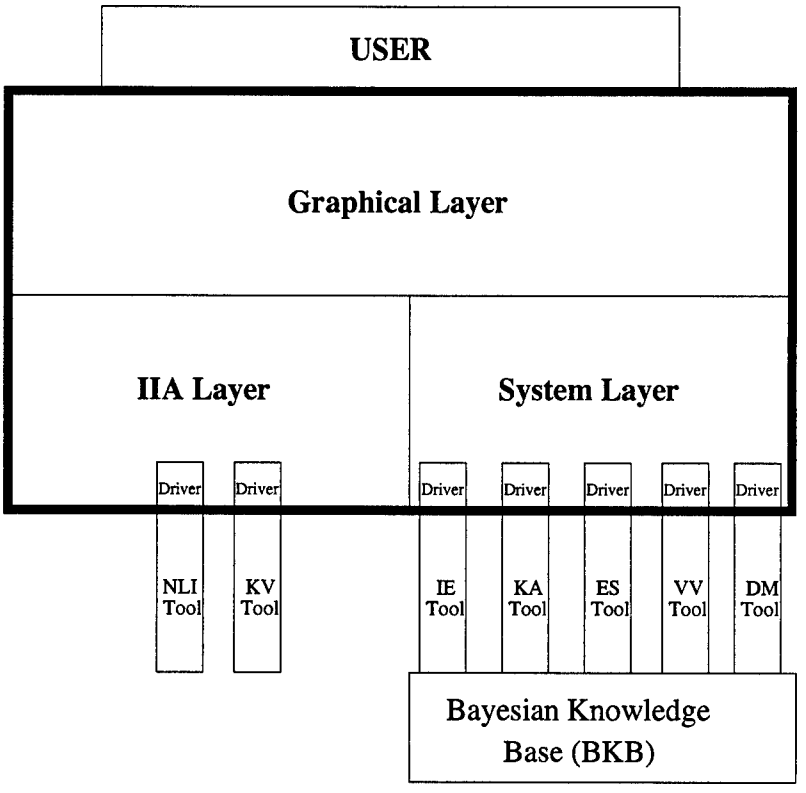

Figure 4.1 **The Layered Interface Architecture.** An illustration that shows the layered architecture of the interface along with the system and communication tools that interact with the layers.

and are the direct line of communication between the user interface and the system tools.

The initial graphical layer designs are performed through storyboards [29, 53, 54]. These storyboards are drawings of interface screens used to prototype the look and feel of the proposed user interface. User reviews are performed to evaluate the prototypes and normalize toward a user interface that supports the interface requirements and follows acceptable standards of graphical interface development. The Motif standard is the chosen design standard for the graphical layer since Motif is the most common graphical user interface standard and the probable choice to use

during implementation. Several rounds of prototyping storyboards defined the final PESKI user interface that are approved by the lead PESKI researcher.

The accumulated design information is transformed into object oriented models suggested by Rumbaugh [72]. Many static and dynamic models of the interface design were created and approved by the lead PESKI researcher. However, research resource limitations and short term deadlines precluded the completion of a comprehensive set of static and dynamic models. Therefore, only a static diagram of the high-level interface objects is presented (see Figure 4.2).

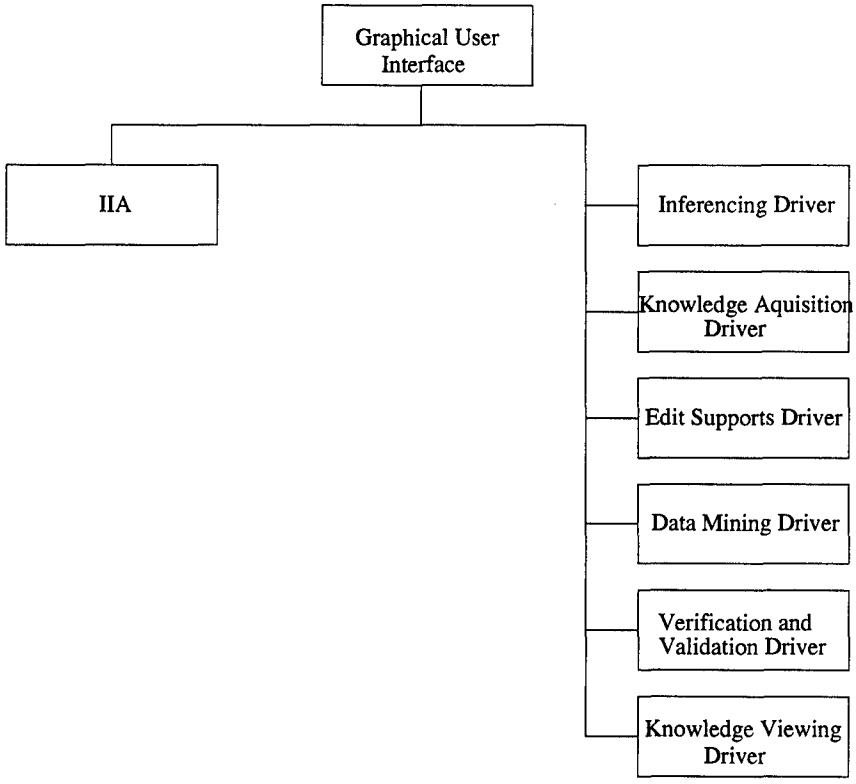

Figure 4.2 **The Interface Architecture Static Model.** An illustration that shows the software objects that make up the interface architecture.

4.4 Designing the Intelligent Interface Agent

As previously mentioned, the IIA layer is responsible for managing the interface's intelligence. The requirements of this element of the intelligent user interface are to model user behavior in order to predict future user behavior or user intent and suggest adaptations. These adaptations are targeted at helping the user utilize the system and adjusting the interface to meet the user's application domain needs.

In order for the intelligent agent to be useful, it must have the ability to reason [24, 12, 61]. In this research, the reasoning capability is enabled by the following: collecting domain metrics, transitioning metrics into a representation, storing information, and inferencing over the stored information. These actions work hand in hand to provide an environment where the user interface can make intelligent decisions. The IIA layer is designed with a knowledge representation, a domain metric protocol, network performance metrics, and a storage protocol that help the layer use the knowledge to meet interface requirements. These elements of the IIA are composed in the objects of a static model(see Figure 4.3).

4.4.1 Knowledge Representation. The basic representation for the IIA's knowledge is a Bayesian-based network called the interface learning network [35, 36, 37, 38]. User behavior is not deterministic, so representing user behavior in an uncertainty-based architecture is appropriate. This representation has the ability to portray a large amount of information based on the collection of only a small number of interface domain metrics, making this representation important for interface reasoning efficiency. There are three types of nodes in the IIA's network: *interface learning nodes*, *interface information nodes*, and *uncertainty support nodes.*

Interface learning nodes are used by the IIA to integrate and store the meaning of collected interface domain metrics (described in Section 4.4.2) into the network. These nodes not only hold a specific semantic meaning but also have a set of probabilistic values attached to the meaning. The semantic meaning of each interface

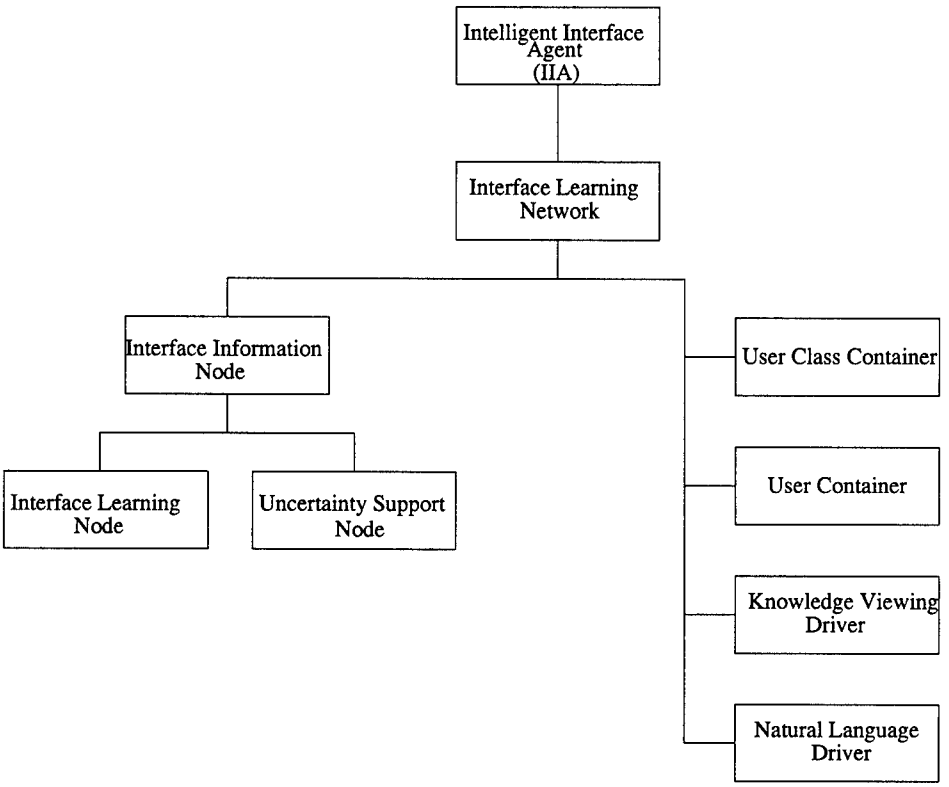

Figure 4.3 **The IIA Static Model.** An illustration that shows the software objects that make up the IIA.

learning node is based on which event the node collects learning, such as a user making a menu choice from an interface window. The actual structure of these nodes consists of the numerator and denominator of a fraction. The specific semantics of each interface learning node combined with its probabilistic values allows the interface learning node to represent degrees of uncertainty in the learned information.

The *interface information nodes* represent the many states of the world within the interface. These nodes utilize interface learning nodes, uncertainty support nodes, and other interface information nodes to determine probabilistic values for the states they represent. While the interface learning nodes are the primary gateway for learning to enter the network, interface information nodes represent the application

of the network's learned knowledge. User interface elicitation of knowledge targets the states represented by the interface information nodes, allowing the user interface to make intelligent decisions concerning potential adaptations.

Finally, the *uncertainty support nodes* store information concerning the uncertainty that the user interface will make a correct decision about a particular interface information node (system state). The structure of these nodes is much like the structure of the interface learning nodes, and there exists exactly one uncertainty support node for each and every interface information node. The probabilities stored in each of these nodes represent all the instances when the interface is correct or incorrect about inferencing over the interface information node it supports. This uncertainty is applied to its parent interface information node to alter its parent's probability when its parent is targeted for knowledge elicitation by the user interface. In this way, the user interface decisions of the future will be affected by its correct and incorrect inferences of the past.

The network design for this thesis tracks three main areas of user intent: BKB file to use, function to use, and communication mode to use. The combination of these three actions, using the topology of the interface learning network, allow for a structure that can make a myriad of suggestions to assist the user(see Figure 4.4). The dark ovals of Figure 4.4 represent interface information nodes while the light ovals represent interface learning nodes. The squares of Figure 4.4 represent uncertainty support nodes and the arcs represent causality.

The topology of the network must be able to alter itself in order for it to better model each user and become more adaptive. In any user interface there are a finite number of states in which the interface can be. These states are defined by the individual states that the interface widgets can be. In a large user interface the number of these states can be enormous, certainly too many to represent simultaneously. Therefore, a user interface intelligence entity must be able to represent a subset of

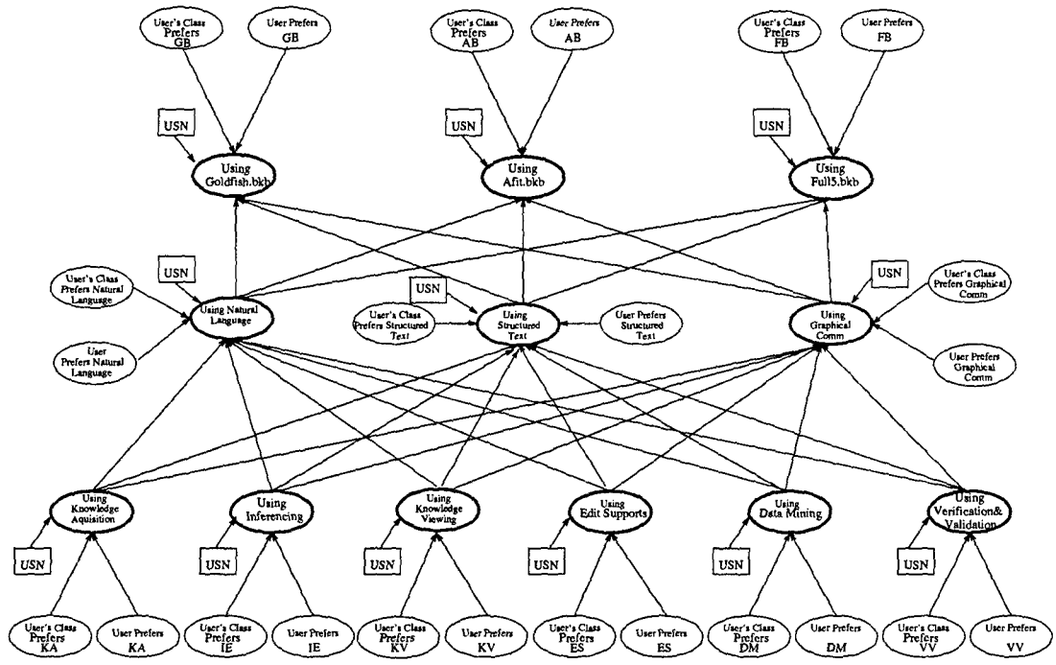

Figure 4.4 **Network Design for the PESKI IIA.** An illustration that shows the network design for the PESKI IIA's application of the interface learning network.

the total number of interface states, and it must represent those states that are most relevant (most useful to assisting the current user of the interface) [16].

Design of topology dynamics occurs at two levels in the IIA. First, add and delete node operations are designed for interface information, uncertainty support, and interface learning nodes. Second, a layer of decision making is added above the interface learning network in order to help the IIA decide when to alter the network. This layer is designed in a basic form, deciding over the domain of BKB filename nodes in the network. When a user selects to use a BKB file that has no node representation in the interface learning network, new nodes and causality connections are created to support the new state. The new node can immediately be used for learning and inferencing. Likewise, when the number of nodes exceed the

size of the relevancy set (the set of relevant interface information nodes), the node with the lowest probability is removed from the network along with any nodes that have dependencies exclusively with the deleted node.

Two important decisions are highlighted when designing the topology dynamics. First, newly created non-class interface learning and uncertainty support nodes begin with probabilities of 0.5, the most uncertain value between 0.0 and 1.0. The newly created class learning nodes receive probabilities based on the class they represent. The second important decision is the limitation on the number of nodes that can be part of the network at any one time. This limitation is set based on computational barriers usually associated with large probabilistic networks.

4.4.2 Domain Metric Collection Protocol. The second element of the IIA's reasoning ability is a domain metric collection protocol for metrics based on the operations being performed on the expert system. These metrics are called *interface domain metrics.* Interface domain metrics can be just about any type of data that a user interface can collect from the application domain or the user. This data includes keystrokes, procedures used to perform tasks, user preferences, and tasks most often performed. The number and type of interface domain metrics collected is solely based on knowledge required for user interface reasoning. Information about the application domain can be acquired from a single interface domain metric or combinations of different types of metrics.

The collected interface domain metrics then need to be transformed into some meaningful information. The information format must be based on one that the user interface requires for making decisions at a later time. This step suggests an intermediate reasoning step that develops a meaning for the metric collected. In the IIA, this step includes incrementing the numerator and denominator elements of interface learning nodes. When the user interface makes a decision, the interface can draw upon knowledge stored in the IIA's network. The architecture of the IIA

allows for the correct return of the metric information stored in the interface learning nodes.

4.4.3 Identification of Network Performance Metrics. Network performance metrics are defined in order to examine the network properties later in testing. There are four metrics adopted for this research: *absolute thrashing*, *class thrashing*, *rate of divergence*, and *rate of convergence* [21]. The intention of this thesis is to present these initial metrics but not to fully explore them. More discussion of these metrics are made in Chapters VI and VIII of this thesis.

Absolute thrashing characterizes an interface information node that, due to perceived user behavior, continually moves in and out of the relevancy set. This occurs when the IIA builds a new interface information node for a newly captured user action, and then the user avoids that action for a period of time. Eventually the node's probability degrades as other actions are made, leading to the node leaving the relevancy set and being deleted by the IIA. It is important to avoid this sequence occurring continually to a single interface information node since every time the node falls out of the relevancy set, its probabilities are lost. Thus, the long term relevancy of the node is lost.

User thrashing characterizes extreme shifts in the probabilities of interface information nodes in the relevancy set. These shifts can indicate a user who can't make up their mind or the inability of the network to accurately predict user intent. Collection of this metric can support dynamic network adaptation decisions of the IIA.

Rate of divergence is the rate of probability change of an interface information node that pulls the node away from the fringes of the relevancy set. *Rate of convergence* is the rate of probability change of an interface information node that pushes the node toward the fringes of relevancy set. Both of these metrics present

trend changes in node probabilities that can also assist the IIA in making dynamic network adaptation decisions.

4.4.4 Storage Protocol. The key to the storage protocol is the *network implementation file.* This file is a representation of the network that can efficiently be stored remotely. The network uses the network implementation file as a roadmap for instantiating network nodes, setting up the parent-child relationships (causalities) between the nodes, and initializing the node probabilities. Every user of PESKI has their own network implementation file.

Every network node is designed to store their probabilities in their instantiated objects at runtime. When a user exits the system, their probabilities must be stored off line in their network implementation file for later use. The user container tracks all network implementation files and manages the file storage and retrieval operations.

The user class container tracks probabilities for interface learning nodes that are designed as class nodes for the four class types: application user, application expert, knowledge engineer, and computer scientist. When a particular user's class learning node receives metrics during system use, the class node is updated for that user's class.

V. *Implementation of the Intelligent User Interface*

Examination of available computing resources led to implementation on a Sun Sparcstation 5 using the C++ programming language with the GCC version 2.7.2 compiler. C++ is the language of choice since all the system tools are written in C++ and Motif, the standard of choice for this interface, is written in C++.

5.1 Implementation of the Interface

Research into automated GUI builders led to the purchase of SPARCworks/Visual [4], a Motif/OSF standard GUI building tool developed by Sun Microsystems. SPARCworks/Visual offers a variety of Motif widgets and allows for quick, object-oriented prototyping and generation of graphical interface designs. The interface storyboard prototypes were converted into a working user interface using the SPARCworks/Visual tools. Additional reviews and prototyping were accomplished on SPARCworks/Visual until the lead PESKI researcher and system tool developers approved the design. C++ code was then generated automatically by the SPARCworks/Visual tool, creating the graphical layer of the design.

The implemented user interface is a multimodal graphical interface that supports structured text and graphical manipulation communication modes. It also provides an interface for a natural language communication mode, however, no natural language processing engine has been developed. Finally, the interface offers fully interactive help through the use of an embedded Netscape [7] application. See Appendix A for Netscape access information.

The structured test communication mode consists of over 700 Motif widgets organized into 8 windows in accordance with the interface storyboards. These allow user access to all expert system functions, including login, using textual representation of information mixed with graphical controls such as buttons and menus.

Graphical communication for the intelligent user interface is provided through a tool called daVinci [32]. This powerful and versatile graph drawing program allows for quick prototyping of graphical manipulations, providing objects and object connectors on a drawing window. These objects can be displayed and manipulated by passing script commands to daVinci and receiving responses back from daVinci. See Appendix A for basic details on access to and use of daVinci.

Each system function has its own graphical communication screen. These screens are accessed from the main PESKI window or from menu choices on each of the structured text screens. All graphical communication screens provide basic viewing of BKB components and states. Full graphical manipulation is not incorporated on all graphical communication screens due to focus of research resources on higher priority subjects. However, the base implementation presented so far does provide the larger test domain for the IIA and proves graphical communication can be implemented. Further implementation of graphical communication is discussed in Chapter VIII.

5.2 Implementation of the IIA

The IIA layer of the design is implemented by converting the previously depicted object classes into C++ programming code. A Bayesian network computing tool called CaBeN [28] is used for inferencing over the interface learning network. This tool is a collection of Bayesian network computing algorithms such as Brute-Force, Logic Sampling, and Chavez. Many of these are stochastic algorithms so they return statistical approximations. However, these stochastic algorithms return answers that are accurate enough for use by the IIA.

5.3 System Integration

Integration of the intelligent user interface occurred once all the individual layers of the design were implemented and tested. The integration began by merging

the system layer and the graphical layer. This merger allowed for testing of the integration as well as testing the interface's ability to access all the system functionalities. The IIA layer was then merged into the system in preparation for testing of the interface intelligence. See Appendix A for information on how to access the final, integrated version of PESKI.

5.4 Communication Between the IIA and the Interface

The communication between the physical user interface and the IIA is implemented to ensure efficient passage of data between the physical user interface and the IIA. There are two types of communication implemented: *learning* and *suggestion.*

Learning communication is enacted when an event occurs in the user interface that the interface recognizes as important to the IIA. These events include the loading of a BKB filename, starting a PESKI tool, and changing communication modes. When such an event occurs, the interface notifies the IIA by calling an IIA method. The IIA interprets the learned event and transfers the learning to the appropriate interface learning node.

Suggestion communication is enacted when an event occurs in the user interface that the interface recognizes as needing a suggestion. The only event implemented currently in PESKI is needing a suggestion at system login. When such an event occurs, the interface notifies the IIA by calling an IIA method. The IIA develops the appropriate suggestion and sends it to the interface for presentation to the user. If the suggestion is accepted by user, the interface carries out the suggestion.

The basic suggestion presentation form is a dialogue window with a primary and alternate suggestion. Each of the two suggestions can hold any number of subsuggestions. For example, the first suggestion can be the most probable system function that the user will want and the second suggestion can be the second most probable system function. Further, the first suggestion can be the most probable system function and the most probable BKB file, and the second suggestion could

be the second most probable system function and BKB file. The combinations of suggestion can vary, and the testing in Chapter VI will examine the performance of some combinations.

VI. Testing the IIA

There are two basic types of testing performed for this research: mathematical soundness and usability. Mathematical soundness testing is performed to ensure the complex mathematics that govern the use of probability distributions are followed. Usability testing explores the usefulness of the research product to real users. Together, these testing forms support the claim that the goals of this research are met.

6.1 Mathematical Soundness Testing

Testing for the accuracy of combining probabilities is based on the premises of Bayes theorem [68, 26]. In the interface learning network there are two possible cases of node configurations that can occur. In the simple case, an interface information node is supported by one uncertainty support node and one or more interface learning nodes. The second, more complex, case includes an interface information node that is supported by one or more interface information nodes, one uncertainty support node, and one of more interface learning nodes.

Each of the two types are tested in three phases. First, the probabilities are combined by hand using accepted mathematical techniques. Second, the same combination of nodes is programmed into the interface learning network and the probabilities are combined using the network's computational facilities. Third, the results of the hand calculations are compared with the results of the network's computations.

6.1.1 Computational Accuracy: Simple Case. This example of a simple network demonstrates how the network learns and how the learned data can be used to create a probability for a possible state [38]. Figure 6.1 depicts the network used in this example. Notice, there is only one interface information node, User is Using Graphical Communication (UGC). There is also one supporting uncertainty support

node, Uncertainty User is Using Graphical Communication (UUGC). Finally, there are two interface learning nodes, User's Class Prefers Graphical Communication (CPGC) and User Prefers Graphical Communication (UPGC).

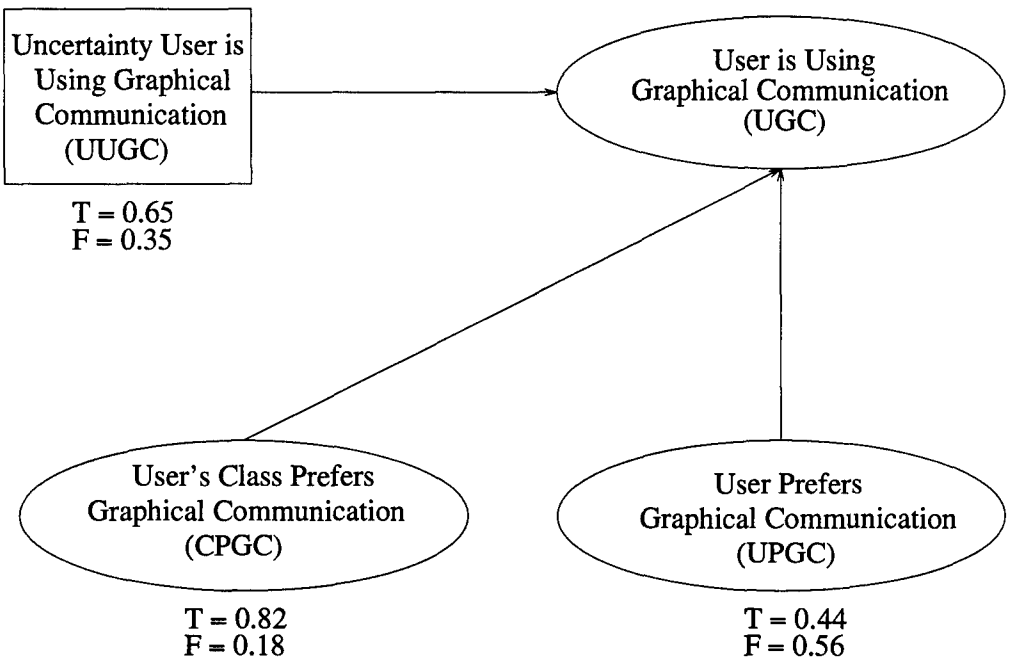

Figure 6.1 **Computational Accuracy: Simple Case Example.** An illustration of the network design for an example that shows how learning and inferencing is performed on an interface learning network.

For this example, a user, login TOM, has logged onto PESKI. The interface learning network recovers all the learned data about TOM from storage and sends the data to the appropriate interface learning nodes and uncertainty support nodes in the network. With the network loaded, TOM begins to use PESKI. As TOM performs actions through the interface, the interface records TOM's behavior by calling the learning method of the IIA, which in turn updates the nodes related to TOM's behavior. For example, in Figure 6.1, if TOM chooses to use graphical communication from the communication mode menu of the interface, the interface

will call the update data methods for the interface learning nodes CPGC and UPGC. Thus, TOM's behavior is captured.

Later, if the user interface wants to predict what communication mode TOM will chose, the interface will query the UGC interface information node, calling the node's compute probability method. This method will then combine the probabilities of interface learning nodes CPGC and UPGC and the uncertainty support node UUGC.

The probabilities are combined using the following method [68, 26]. First, a truth table is constructed that lists all the possible combinations of the truthfulness of the interface learning nodes. Therefore,

$$P(UGC = T|\ CPGC = T,\ UPGC = T) := 1.00,$$
$$P(UGC = T|\ CPGC = T,\ UPGC = F) := 0.65,$$
$$P(UGC = T|\ CPGC = F,\ UPGC = T) := 0.65, and$$
$$P(UGC = T|\ CPGC = F,\ UPGC = F) := 0.00.$$

Notice, if the probabilities that CPGC and UPGC are both true then the probability of UGC being true is 1.00, and if the probabilities that CPGC and UPGC are both false then the probability of UGC being true is 0.00. The usefulness of the uncertainty support node UUGC comes into play when the probability of an interface information node is not absolute. In these cases, the uncertainty of the truthfulness of the node must support the interface information node. In this case, UUGC gives the value of 0.65 to the conditional probabilities of the uncertain values of the truth table.

Once the truth table is constructed, the probabilities may be combined using Bayes theorem:

$$\begin{aligned} P(UGC = T) = \ & P(UGC,\ CPGC,\ UPGC) \\ & +(UGC,\ \neg CPGC,\ UPGC) \\ & +(UGC,\ CPGC,\ \neg UPGC) \\ & +(UGC,\ \neg CPGC,\ \neg UPGC) \end{aligned}$$

$$
\begin{aligned}
P(UGC = T) = \quad & 1.00 * 0.82 * 0.44 \\
+ \; & 0.65 * 0.18 * 0.44 \\
+ \; & 0.65 * 0.82 * 0.56 \\
+ \; & 0.00 * 0.18 * 0.56
\end{aligned}
$$

Therefore, (UGC=T) = 0.7108 or 71%, the same result produced by the implemented version of the interface learning network. Given this result, the user interface has acquired a mathematically sound method to capture user behavior and then convert it into a representation so the user interface may reason about future user intent.

6.1.2 Computational Accuracy: Complex Case. This example covers a more complex network, demonstrating how probabilities from one interface information node can be used to support another one [38]. Figure 6.2 depicts the network for this example. This example expands the simple case by adding a new interface information node, User is Using Knowledge Acquisition (UKA), that is supported by one new interface learning node, User Prefers Knowledge Acquisition (UPKA), and one of the previous interface learning nodes, User Prefers Graphical Communication(UPGC). The new uncertainty support node that supports UKA isUncertainty User is Using Knowledge Acquisition (UUKA).

For this example, a user, login JANE, has logged onto PESKI. The network is initiated much like the scenario in the simple case. Later, if the user interface wants to predict what interface tool JANE will chose, the interface will query the UKA interface information node, calling the node's compute probability method. This method will then combine the probabilities of UKA's child nodes.

The probabilities are combined in the following way. First, a truth table is constructed that lists all possible combinations of the truthfulness of the child nodes. Therefore,

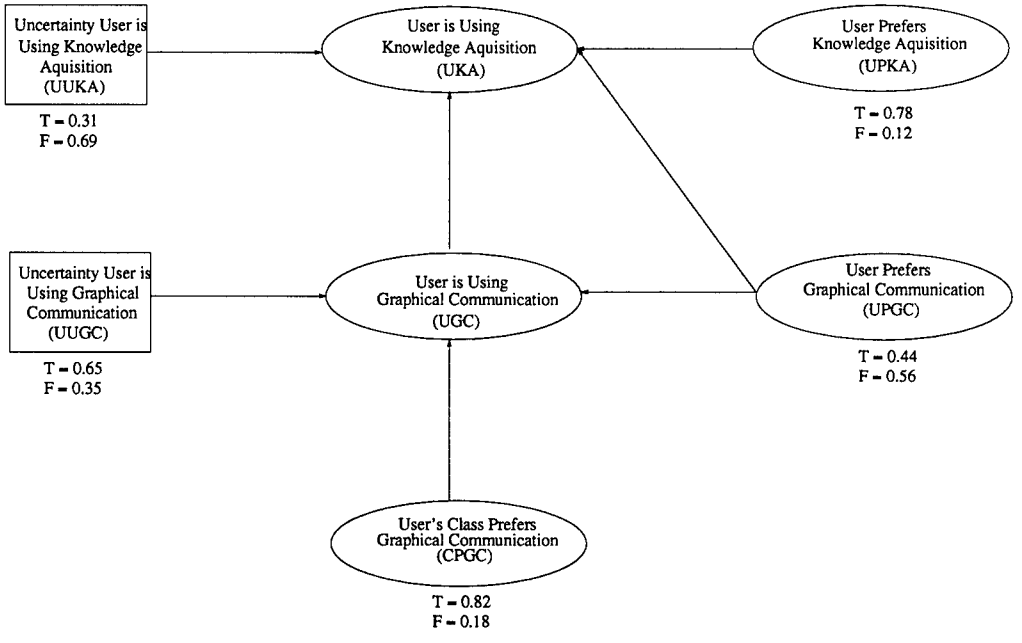

Figure 6.2 **Computational Accuracy: Complex Case Example.** An illustration of the network design for an example that shows how learning and inferencing is performed on an interface learning network.

$$P(UKA = T|\ CPGC = T,\ UPGC = T,\ UGC = T,\ UPKA = T) := 1.00,$$
$$P(UKA = T|\ CPGC = F,\ UPGC = T,\ UGC = T,\ UPKA = T) := 0.31,$$
$$\ldots,$$
$$P(UKA = T|\ CPGC = T,\ UPGC = F,\ UGC = F,\ UPKA = F) := 0.31, \textit{and}$$
$$P(UKA = T|\ CPGC = F,\ UPGC = F,\ UGC = F,\ UPKA = F) := 0.00.$$

Once the truth table is constructed, the probabilities may be combined using Bayes theorem:

$$
\begin{aligned}
P(UKA=T) = \; & P(UKA, \; CPGC, \; UPGC \; UGC \; UPKA) \\
& +(UKA, \; \neg CPGC, \; UPGC \; UGC \; UPKA) \\
& \ldots, \\
& +(UKA, \; CPGC, \; \neg UPGC \; \neg UGC \; \neg UPKA) \\
& +(UKA, \; \neg CPGC, \; \neg UPGC \; \neg UGC \; \neg UPKA)
\end{aligned}
$$

$$
\begin{aligned}
P(UKA=T) = \; & 1.00 * 0.82 * 0.44 * 0.71 * 0.78 \\
& +0.31 * 0.18 * 0.44 * 0.71 * 0.78 \\
& \ldots, \\
& +0.31 * 0.82 * 0.56 * 0.29 * 0.12 \\
& +0.00 * 0.18 * 0.56 * 0.29 * 0.12
\end{aligned}
$$

Therefore, (UGC=T) = 0.5023 or 50%, the same result produced by the implemented version of the interface learning network. As in the simple case, the user interface has demonstrated a mathematically sound method to capture user behavior and then convert it into a representation. This example also shows how explosive computations can be when the network size is expanded.

6.2 Usability Testing

There are five general tests that are used in this research to indicate a successfully usable interface intelligence. The first test is a collection of physical work requirements that quantify procedures the user must follow to get work done. How positively or negatively the user feels about using the interface intelligence is captured in the second test. The third test measures responsiveness burdens the intelligence places on the interface. The final test examined the accuracy of the user model of the intelligence, or in other words, how closely the model actually represents user intent.

6.2.1 Physical Work Requirements. Collecting the physical work a user is required to do is one way to evaluate the usefulness of the interface intelligence.

Physical work requirements such as keystrokes, menu selections, reading, and button presses are collected for a user utilizing the interface intelligence. Care must be taken when drawing conclusions from physical work requirements since this data does not form a complete picture of interface usability.

The current implementation of the IIA makes suggestions pertaining to what system function, communication mode, and BKB file the user wants to access at system startup. Therefore, this test concentrates on physical work required of the user if the user starts these choices themselves versus the physical work required if the user interacts with the IIA to make these choices.

Appendix B shows the data collected to test the usefulness of the IIA. These results clearly show that using the IIA's suggestions yields a considerable savings in physical work for the user. This data can be compared with the data from user model accuracy tests to ensure a true cost savings in physical work over time. These results also show that over time the user receives a work savings when using the IIA instead of making startup choices manually.

6.2.2 Acceptance of the IIA. User acceptance data is collected by exposing a number of users to the interface intelligence and eliciting user opinion on a written survey. This survey is a carefully constructed list of instructions and questions that guides the user though the IIA's capabilities and require exact and free-form responses from the user concerning these capabilities. The survey is clear, consistent, and precise. The survey used for this research is found in Appendix C.

Collecting data that indicates user acceptance of an intelligent user interface is difficult. User acceptance is affected by factors such as user's computing experience, prejudices, and constantly changing moods. Therefore, like the collection of physical work requirements, care is taken when drawing conclusion solely from user acceptance results.

Appendix D lists the results of a small user acceptance study for the IIA. The sample size is five users of somewhat high computer experience. The test results are divided into three general areas: timeliness of operations, complexity of operations, and usefulness of the IIA.

Users were generally satisfied with the timeliness of operations, although they seem to find the automatic operations performed by the IIA slightly slower than performing the same operations manually. This contradicts the responsiveness testing results from Appendix E that show faster performance using the IIA's suggestions.

The results of the complexity evaluation are mixed. Users seemed to find the single suggestion to loading a BKB file less complex than loading the BKB file manually. This result is supported by the physical work requirements for BKB loads found in Appendix B. However, users found the double suggestion of system function and communication mode more confusing than manually choosing the system function and communication mode from the main window. This result is somewhat supported by the fact that the work requirements for manually selecting the system function and communication mode are low.

The usefulness of the IIA's suggestions are as expected. Users generally found the IIA to be useful, although these results are most probably influenced by the results for user opinion on timeliness and complexity. A more indepth user acceptance study is desired to collect long term opinions of the IIA from many users using PESKI to perform real tasks (see Chapter VIII).

6.2.3 Responsiveness of the IIA. Responsiveness of an interface is typically an important criteria for interface users. Therefore, testing the responsiveness of the interface, particularly the effect intelligence has on interface responsiveness, is a collection of user opinions and empirical data. The user opinions are collect in the same manner described in the user acceptance test description. Empirical data is taken by collecting real time data during interface functions that are influenced by

the interface's intelligent structures. Together, this data can give a good picture as to the acceptability of the intelligent user interface's responsiveness.

Appendix E lists the results of the responsiveness study for the IIA. The real time data indicates the current implementation of the IIA creates some user noticeable pauses. The noticeable pauses are created mainly by update calls to CaBeN and execution calls for daVinci. The user acceptance study shows that users found the pauses noticeable but acceptable.

6.2.4 Accuracy of the User Model. A study of the IIA's ability to predict future user behavior is necessary in order to determine if the IIA accurately models user intent. This can be accomplished by observing the dynamics of the agent's suggestion generation capabilities when given a set of test cases that mimic user behavior [38]. There are three test cases used for this research to explore the accuracy of the user model: *single focus*, *double suggestion*, and *triple suggestion.*

6.2.4.1 Overcoming Evidence to Adapt Single Focus Case. In this case the probabilities of two interface information nodes, Using Data Mining (UDM) and Using Inference Engine (UIE), are tracked through the case of a user changing their mind. The user starts PESKI for the first 15 times with the intention of using the Data Mining function. This user accepts any suggestion for starting Data Mining and rejects any suggestion that doesn't suggest Data Mining. After the initial 15 times the user switches their preference to the Inference Engine, accepting or rejecting suggestions based on this new preference. For this case, only the function suggestion is evaluated and all other elements of each suggestion (BKB filename and communication mode) are ignored.

The results of this test are shown in Figure 6.3. These results clearly show the IIA's ability to adapt to the user's change in preference. In this case, the IIA is able to generate a correct suggestion in only two iterations from the change in user behavior.

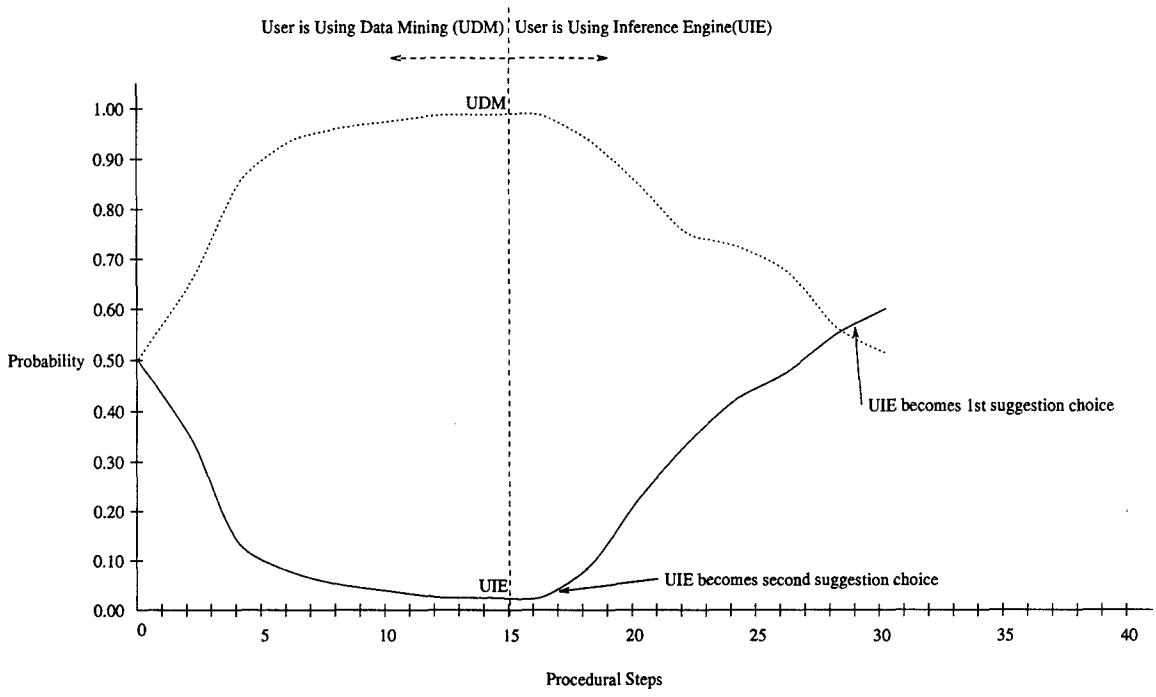

Figure 6.3 **Overcoming Evidence to Adapt Single Focus Case.** An illustration that shows how quickly the network probabilities change when a user builds evidence towards one single state choice and then changes their choice.

6.2.4.2 Overcoming Evidence to Adapt Double Suggestion Case. This test case differs from the last in that the focus of the user is on a combination of suggestions including BKB filename, system function, and communication mode. At system startup, two BKB filenames are suggested. Once one is chosen or both rejected another two suggestions appear, giving suggested function and communication mode combinations. This combination or *double* suggestion must be completely true (both parts) in order for the user to accept it.

In this case the probabilities of six interface information nodes are tracked: Using Knowledge Acquisition (UKA), Using Inference Engine (UIE), Using Text

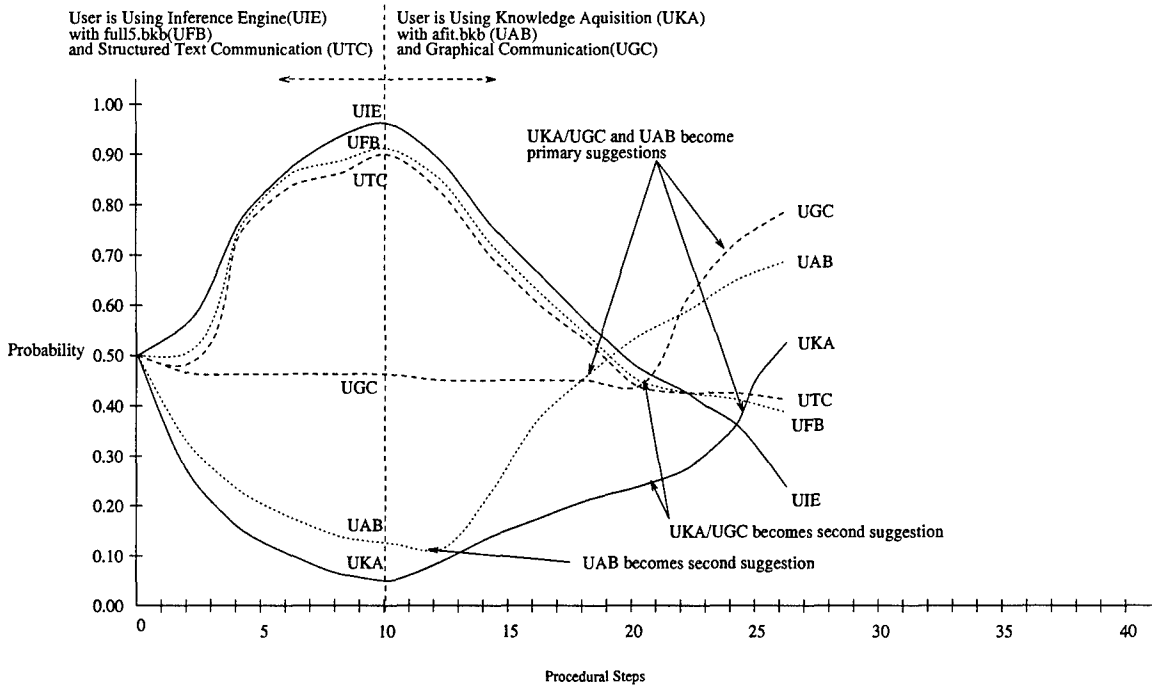

Figure 6.4 **Overcoming Evidence to Adapt Double Suggestion Case.** An illustration that shows how quickly the network probabilities change when a user builds evidence towards one double and one single state choice and then changes their choice.

Communication (UTC), Using Graphical Communication (UGC), Using Full5.bkb (UFB), and Using Afit.bkb (UAB). The user starts PESKI for the first 10 times with the intention of using the Inference Engine function with Text Communication and Full5.bkb. This user accepts any suggestion that is completely true and rejects any suggestion that is not completely true. After the initial 10 times the user switches their preference to the Knowledge Acquisition function with Graphical Communication and Afit.bkb, accepting or suggesting behavior based on these new preferences.

The results of this test are shown in Figure 6.4. These results clearly show the IIA's ability to adapt to the user's change in preferences. It should also be noted how the causalities within the network (see Figure 4.4) have interesting effects on the nodes tracked in this case, such as the behaviour of UGC versus the behavior of UAB and UKA. Also, the acceptance and rejection of suggestions, especially the rejection of suggestions that are partially but not fully correct, have an interesting affect on the probability distribution throughout the network. This fact is the reason for UAB's rapid increase in probability after the 17th step while UKA only maintains a steady rise in probability.

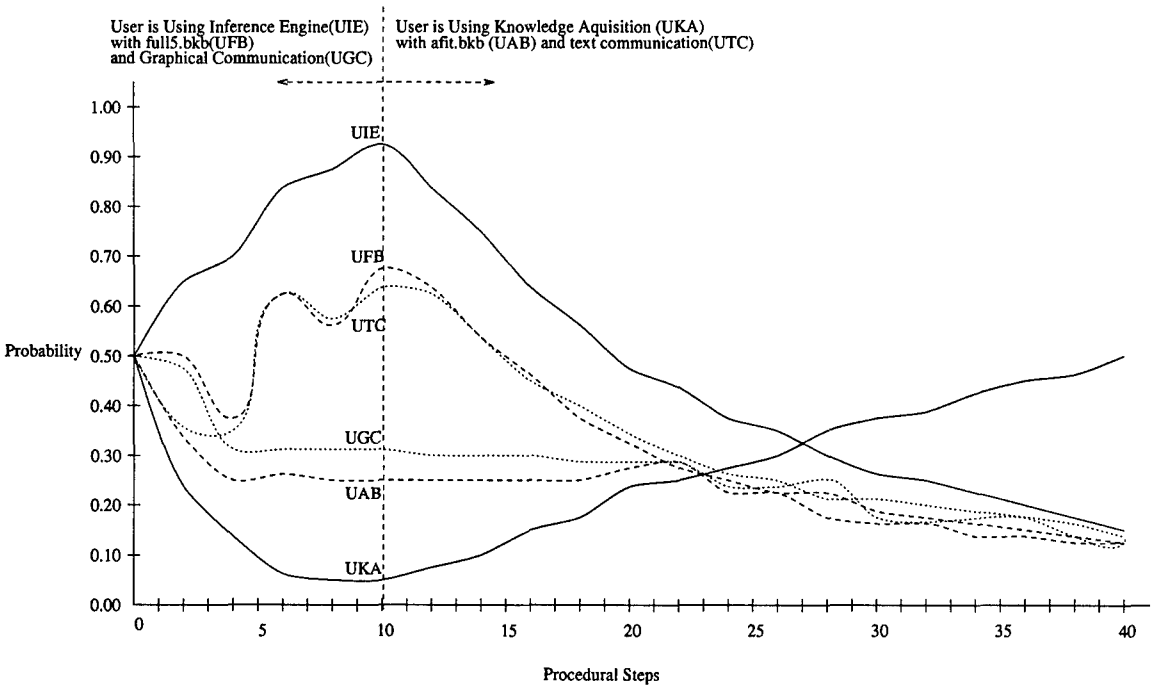

Figure 6.5 **Overcoming Evidence to Adapt Triple Suggestion Case.** An illustration that shows how quickly the network probabilities change when a user builds evidence towards one triple state choice and then changes their choice.

6.2.4.3 Overcoming Evidence to Adapt Triple Suggestion Case. This case is exactly like the previous case except that the suggestion generation method is different. At startup, the user is only provided two suggestions, each a combination of system function, communication mode, and BKB filename. As with the previous combination suggestion, all parts of this *triple* suggestion must be true for the user to accept the entire suggestion.

The results of this test are shown in Figure 6.5. These results show a failure of the IIA to adapt to the user's new behavior. This failure is due to the practice of rejecting triple suggestions that aren't totally true. This creates what appears to be user thrashing, but the erratic behavior is really a failure in the application of the suggestion method. This highlights the need to use the IIA carefully when applying it to a target system.

VII. Results and Conclusions

The evaluation of this research focuses on three areas. First, the results of this research are compared with the original goals to conclude the success or failure of meeting the goals. Second, strengths of this research are examined to highlight the impacts of this research on the study of interface intelligence. Third, the weaknesses are uncovered that set the stage for the future research described in Chapter VIII.

7.1 Accomplishment of Research Goals

The four initial goals described in Chapter 1 of this thesis are designed as a roadmap toward performing successful research. Each of these goals are now examined in detail to determine if the goals have been met, and if not, to what extent they have been met.

7.1.1 Modeling User Intent. The testing performed in Chapter VI, particularly in Section 6.2.4, shows that the intelligent user interface can adapt its suggestion based on changes in the user's behavior. This supports the claim that the IIA does model user intent. However, the test results also show the changes occur only gradually, sometimes too slowly to claim the IIA perfectly models user intent. For example, the IIA fails to recognize trends in user behavior that could be used by the IIA to alter the user's profile more quickly. The test case from Section 6.2.4.3 also highlights that misuse of the IIA can lead to an inaccurate model. Chapter VIII discusses some future research that can address these problems.

7.1.2 Complexity Abstraction. Whether or not this goal is met can be argued. On one hand, the current implementation of the interface intelligence lets the user accept suggestions to perform tasks and save the user work, as seen in Section 6.2.1. For example, a user who accepts a suggestion to load a particular BKB file is saved from the work of searching for the file through directories and then

loading it. In this way, the IIA does provide some complexity abstraction. Yet, the IIA's suggestions don't currently apply to the real complex operations of PESKI, such as knowledge acquisition tasks. Once the IIA is applied to a wider range of complex system functions, a true evaluation of complexity abstraction can be taken.

7.1.3 Maintenance of Generic Nature. The intelligent user interface does meet the goal of maintaining the generic nature of PESKI. The physical user interface is modeled to support PESKI's tools but there are no signs of a particular application domain in the final interface. The suggestions presented by the IIA currently pertain to the system's functions, communication modes, and BKB filenames, all of which are independent of the application where PESKI is used.

7.1.4 Viability of Bayesian Interface Intelligence. The testing performed in Chapter VI, when observed as a whole, shows great promise for using Bayesian techniques for interface intelligence. The mathematical accuracy and ability of the IIA to model user intent prove the ability to apply Bayesian techniques to the study of intelligent user interfaces. The responsiveness, work, and user preference studies show a technique that is acceptable, but needs to be refined in responsiveness. Finally, the ability to capture uncertainty through Bayesian techniques has immense value to the accuracy and dynamics of IIA's predictions.

7.2 Strengths of the Research

There are three major strengths to this research: mathematical accuracy for capturing uncertainty, adaptability, and foundation for Bayesian-based interface intelligence. Together, these strengths support the claim that this research is an important, useful contribution to the study of intelligent user interfaces.

7.2.1 Mathematical Accuracy for Capturing Uncertainty. The Bayesian methods used for manipulating the probabilities of the interface learning network

have a solid basis in proven mathematics. Using these proven mathematical foundations, uncertainty is captured and utilized effectively to perform predictions of user intent. This strength shows a sincere effort to avoid ad-hoc methods for interface intelligence such as statistical-based limits and certainty factors.

7.2.2 Adaptability. The ability of the IIA to adapt suggestions is very dynamic and strongly tied to mathematically correct probabilistic changes. The current implementation is generic enough in nature to allow application of the IIAs adaptability to many interface domains and users. This strength shows the usefulness of applying the IIA to other systems that require interface intelligence.

7.2.3 Foundation for Bayesian-Based Interface Intelligence. As shown in Chapter VIII, this research provides a firm foundation to spawn more studies into Bayesian based interface intelligence. This research area is brand new to the Air Force Institute of Technology (AFIT), and this thesis has ensured that Bayesian-based interface intelligence is now a continuing part of the AFIT research program. Also, the development of the IIA's mathematically and semantically sound representation has already drawn interest and support from editors of notable scientific publications [21, 35].

7.3 Weaknesses of the Research

The four main weaknesses of this research are computational barriers, verification of user intent models, the overlooking of key indicators, and the additions to the methodology. Overcoming these main barriers are the key to the future of this research, as seen in Chapter VIII.

7.3.1 Computational Barriers. The computational problems common to many probabilistic representations is a serious barrier to the scalability and performance of the interface learning network. The current solution found in CaBeN still

has problems with the inaccuracy of stochastic methods and slowness in updating CaBeN's network representation. The computational barrier may not be completely removed but may be pushed back in order to improve the usability of the current IIA representation.

7.3.2 Verification of User Intent Models. Simply running test cases in order to simulate and observe the network's ability to model user intent is not enough to fully verify the accuracy of user intent and usability of the IIA. Full usability studies need to be performed on the IIA in order to truly claim the research goals are met.

7.3.3 Overlooking of Key Indicators. As seen in Chapter VI, Section 6.2.4 the IIA does adapt to changes in user behavior, but those changes are too slow. Further, the IIA misses key indicators, such as the rate of convergence and rate of divergence (discussed in Chapter IV) that show the user has drastically changed their behavior. This miss causes the interface to continue making wrong predications far longer than it should.

7.3.4 Additions to Methodology. The original methodology in Chapter III is a good framework for this research and is key to the overall research success. However, there are many areas of the methodology that have proved to be too scarce in details. For example, Section 3.4 gives some indications on how to go about developing the interface intelligence but lacks details on exactly how to evaluate and choose an appropriate knowledge representation. Identification and detailing of these sparse areas would improve reusability of this methodology for future research (see Chapter VIII).

VIII. Further Research

There are many areas of research that can be spawned from the results shown in this thesis. Addressing these areas will improve the viability of this research and overcome the weaknesses described in Chapter VII. The most important areas for future research are listed in this chapter, although many more may be realized.

8.1 Development of an Interface Intelligence Language

Communication protocols between the IIA and the graphical user interface became increasingly important in the later phases of this research. One particular problem area was the visibility requirements of the IIA and the graphical user interface to each other. Increasing amounts of visibility between the two entities degraded the portability and modularity of the IIA.

One promising area of research is to define a common interaction language that can be used by the IIA and the graphical user interface [21]. This language can be developed by first producing a finite state machine that models a typical interaction between the two entities. Next, a grammar can be developed that defines the machine. This grammar can then become a common language that the IIA and graphical user interface can use. Transplanting the IIA to another user interface will then be only a matter of integrating the new grammar into the new user interface.

Development of this language can lead to improved portability of the IIA. The production of a portable interface intelligence can be of great use to the scientific community. The availability of a proven intelligence that can easily be tied into any system's user interface will allow the benefits of interface intelligence to be spread to all types of systems. This availability will spur increasing dependency for intelligent assistance and show to the average user that artificial intelligence can be useful.

8.2 Meta-Levels of Interface Learning

The ability of the IIA's network to model user behavior can be expanded by designing the interface to understand user behavior. For example, if the interface measures patterns of indicator swings and stores those patterns for a particular user, the interface may begin to classify those patterns. The interface may then be able to assign patterns to user traits, such as moods. The incorporation of temporal reasoning into this representation would allow the interface to predict user traits based on the stored patterns [41].

Research should be started into efficient ways for the meta-level of intelligence to recognize when it is truely correct and incorrect about a prediction. Currently, the IIA only knows it has made an incorrect prediction if the user rejects that prediction. A more sophisticated intelligence may require memory of past state changes to really understand when it is correct and incorrect. Solving this problem is key to realizing the real value of the uncertainty support nodes, since their values are updated from correct and incorrect predictions.

Currently, the IIA possesses some element of meta-learning, evident in the uncertainty support nodes. The IIA can be enhanced if it is able to interpret why it makes bad predictions. Changes in probabilities could be used as key indicators to the IIA. These indicators can be used to determine that a problem exists and to find ways to solve the problem. For example, if the IIA sees wide swings in interface use from these indicators, it may be able to make more dynamic predictions. Taking this idea a step further, once the IIA recognizes a cause of failure, the IIA can dynamically alter the relationships in the network to reduce the chances of reoccurrence.

8.3 Computational Efficiency

One area of further research is to find ways to make the network computations more efficient. Increased efficiency would expand the maximum size of networks considered to be computationally reasonable. This can be accomplished by listing

and prioritizing potential user and interface behavior in order to find the relevant behavior to model. With the limit on network size, this study could point to behavior that is not relevant to IIA predictions.

Computational efficiency gains can also be made by adopting a better technique for network inferencing. The current use of CaBeN has proven to be effective, but CaBeN imposes responsiveness burdens on the IIA. These burdens can be removed by implementing an efficient stochastic algorithm directly into the IIA. This implementation has the potential to dramatically improve responsiveness and usability of the IIA.

8.4 Refinement of the PESKI User Interface

Further refinement of the PESKI user interface's capabilities with graphical manipulation, natural language communication [39], and dynamic dialogue production [70] will provide a richer environment for exercising the IIA. Graphical manipulation can be refined by implementing additional graphical control in daVinci windows, such as moving, adding, and deleting BKB elements. The introduction of a natural language parser and standardized grammar will allow full natural language capabilities to be enacted. Finally, the addition of dynamic dialogue for PESKI's tool operations will allow greater user interaction with the tools. These improvements to PESKI will also allow the interface learning network to be expanded in scope, providing numerous and varied suggestion opportunities for PESKI users. This, in turn, will lead to better testing and understanding of the IIA's capabilities.

8.5 Connections Between Network and Cognitive Models

Tying similarities between aspects of the interface learning network model to already proven cognitive models of user behavior can significantly strengthen the premises of this research. User models can be described by graphically capturing user behaviour, such as the graphs presented in Chapter VI. These graphical repre-

sentations can be compared with the results of of user modeling research, particularly with intelligent tutoring research [10, 87]. If such ties can be found, they would garner support from cognitive science researchers in the search for more accurate, efficient intelligent user interfaces.

8.6 Development of a Generic Methodology for Interface Intelligence Research

As described in Section 7.3.4, the methodology framework for this research is sound but lacks in some details. Addition research details can be derived that would enhance to completeness of the methodology. Therefore, a future research initiative is to take the base methodology from Chapter III and the uncaptured details in order to create a proven, generic methodology for developing interface intelligence. The additional details should mainly support three areas of the methodology: developing requirements, designing the intelligence, and testing the integrated intelligent user interface.

8.6.1 Additions to Requirements Development Methodology. One action that would greatly enhance the requirements development methodology is to create abstract models of the potential system users during the requirements definition phase. These models should map the facts gathered concerning potential user classes to a model that depicts how that behavior relates to the user interface and the software system. The value of these models would emerge during design of the intelligence structure, providing a firm foundation for designing the knowledge representation and domain metric protocol.

The user models can be complemented by evaluating the precise needs for intelligent intervention between user behavior and system performance. This study should focus on two types of interactions: those where intelligence is required and those where intelligence is beneficial. Interactions where intelligence is required can include those where very complex human-computer interaction takes place, those

that require large amounts of user short-term memory, or those where high user error rates are expected. Interactions that are beneficial to the user can also be explored. These interactions can include those that require numerous, repetitive tasks, those that create an environment conducive to greater user acceptance of the system, and those that reduce the user error rate.

8.6.2 Additions to Intelligence Design Methodology. There are three areas of the intelligence design methodology that need to be detailed: knowledge representation development, metric identification, and defining the relevancy set.

The design process for the knowledge representation should include a complete study of available knowledge representations. Consideration must be given to representation traits. The ability to represent uncertainty is an important trait to evaluate. Uncertainty gives the knowledge representation a powerful ability when generating suggestions in the uncertain environment of human behavior prediction. Computational performance is another critical trait to evaluate. The efficiency of learning and inferencing on a knowledge representation can determine if a representation is even usable. Also, scalability of the representation must be considered. This may not be a factor for a small, static system. However, large systems that require dynamic interface intelligence will require a knowledge representation that can grow with a growth in user needs.

Metrics are used in this thesis to define data learned from the application domain and performance of the interface learning network. A generic methodology for developing interface intelligence should include a detailed examination of these two types of metrics. The interface domain metrics should be identified by examining the user models, the target system, and application domain to find specific actions that bind them all in human-computer interaction. Once these binding actions are identified, they must be compared with the chosen knowledge representation to find a mapping of the actions to the representation. The performance metrics should

be developed to target those areas where use in the knowledge representation affect factors such as scalability, responsiveness, and accuracy. These metrics are not only useful for evaluating the intelligence in testing, see Section 8.6.3, but can be used for the development of meta-levels of intelligence as discussed in Section 8.2.

Formally defining the relevant set data to collect is important to realizing a scalable, usable intelligence. Most system environments are too complex to collect data about everything. Therefore, a generic methodology should include a study to evaluate how much data from the user models is needed to support the intelligent intervention. This must be a balanced between the maximum amount of data that can be managed by the knowledge representation and keeping the representation computationally efficient.

8.6.3 Additions to Testing Methodology. The main addition to the testing methodology is the inclusion of a full range of usability studies. This research performed a small number of usability studies to get a general feel for the usability of the intelligence. However, an inclusive, generic methodology should include a complete usability study. Such a study should target a large subset of users that represent all possible classes and levels of experience for the target system. These users should use the implemented system to perform system operations both with the intelligence active and without the intelligence active. Collection of data for this study should be a combination of system collected data and user surveys.

Bibliography

1. *High Performance Computing and Communications FY 1994 Blue Book.* URL: http://www.hpcc.gov/blue94/section.3.4.html: National Coordination Office for High Performance Computing and Communications, 1994.
2. *High Performance Computing and Communications FY 1995 Blue Book.* URL: http://www.hpcc.gov/blue95/section.2.5.html: National Coordination Office for High Performance Computing and Communications, 1995.
3. *High Performance Computing and Communications FY 1995 Implementation Plan.* URL: http://www.hpcc.gov/reports/index.html: National Coordination Office for High Performance Computing and Communications, 1995.
4. *SPARCworks/Visual User's Guide: Version 1.1.* Mountain View, CA: Sun Microsystems, Inc., 1995.
5. *High Performance Computing and Communications FY 1994 Blue Book.* URL: http://www.hpcc.gov/reports/index.html: National Coordination Office for High Performance Computing and Communications, 1996.
6. *High Performance Computing and Communications FY 1996 Blue Book.* URL: http://www.hpcc.gov/blue96/section.2.4.html: National Coordination Office for High Performance Computing and Communications, 1996.
7. *The Netscape WWW Homepage.* URL: http://home.netscape.com/: Netscape Communications Corporation, 1996.
8. Aliferis, Constantin F. and Gregory F. Cooper. "A Structurally and Temporally Extended Bayesian Belief Network Model: Definitions, Properties, and Modeling Techniques." *In the Proceedings of the Twelfth Conference for Uncertainty in Artificial Intelligence.* 1996.
9. Allen, James. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc., 1995.
10. Ardissono, Liliana and Dario Sestero. "Using Dynamic User Models in the Recognition of the Plans of the User," *User Modeling and User Adapted Interaction*, *5*:157–190 (1996).
11. Avouris, Nicholas M. and Sandra Finotti. "User Interface Design to Expert Systems Based on Hierarchical Spatial Representations," *Expert Systems With Applications*, *6*:109–118 (1993).
12. Baecker, Ronald M. and others. *Readings in Human-Computer Interaction: Toward the Year 2000, Second Edition.* San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.
13. Banks, Darwyn O. *Acquiring Consistent Knowledge for Bayesian Forests.* MS thesis, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, 1995.

14. Bennasat, Izak and Peter Todd. "An experimental investigation of interface design alternatives: icon vs. text and direct manipulation vs. menus," *International Journal of Man-Machine Studies*, *38*:369–402 (1993).

15. Benyon, D. and D. Murray. "Adaptive systems: from intelligent tutoring to autonomous agents," *Knowledge-Based Systems*, *6*:197–219 (1993).

16. Benyon, David, et al. "Computer-aided Adaptation of User Interfaces," *SIGCHI Bulletin*, *26*(11):25–27 (1994).

17. Bernstein, Daniel J. *Using Motif with C++*. New York, NY: SIGS Books, 1995.

18. Borghetti, Brett, et al. "Inferencing Over Incomplete Solution Spaces with Genetic Algorithms for Probabilistic Reasoning." *In M. Gasser (Ed.), Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference. URL http://www.cs.indiana.edu/event/maics96/Proceedings/Borghetti.html*. 1996.

19. Bos, Edwin, et al. "EDWARD: full integration of language and action in a multimodal user interface," *International Journal of Human-Computer Studies*, *40*:473–495 (1994).

20. Boutilier, Craig and others. "Context-Specific Independence in Bayesian Networks." *In the Proceedings of the Twelfth Conference for Uncertainty in Artificial Intelligence*. 1996.

21. Brown, Scott M., et al. "A Dynamic Bayesian Intelligent Interface Agent." *Submitted to 1997 Florida Artificial Intelligence Research Symposium*. 1996.

22. Burnell, Lisa J. "Intelligent Software Maintenance," *PC AI*, *July/August*:16–21 (1996).

23. Carolis, Berardina De and Fiorella de Rosis. "Modelling Adaptive Interaction of OPADE by Petri Nets," *SIGCHI Bulletin*, *26*(2):48–52 (April 1994).

24. Cesta, Amedeo and Daniela DAloisi. "Building Interfaces as Personal Assistants," *SIGCHI Bulletin*, *28*(3):108–113 (1996).

25. Chappel, H., et al. "Engineering User Models to Enhance Multi-Modal Dialogue." *In JA Larson and CA Unger, editors, Engineering for Human Computer Interaction, Elsevier Science Publishers*. 1992.

26. Charniak, Eugene. "Bayesian Networks without Tears," *AI Magazine*, *Winter*:50–63 (1991).

27. Cohen, Philip R. "The Role of Natural Language in a Multimodal Interface." *In the Proceedings of the ACM Symposium on User Interface Software and Technology, UIST'92*. 1992.

28. Cousins, Steve B., et al. *CaBeN: A Collection of Algorithms for Belief Networks*. Technical Report WUCS-91-25, Medical Informatics Computer Systems,

Department of Computer Systems, Washington University, St. Louis, Missouri, 1991.

29. Cox, Kevin and David Walker. *User-Interface Design*. New York: Prentice Hall, 1993.

30. Donskoy, M. V. "An Architecture for Object Oriented User Interfaces for a Model-Based Diagnostic System." *In Lecture Notes in Computer Science, 4th International Conference, EWHCI'94*. 1994.

31. Friedman, Nir and Moises Goldszmidt. "Learning Bayesian Networks with Local Structure." *In the Proceedings of the Twelfth Conference for Uncertainty in Artificial Intelligence*. 1996.

32. Frohlich, Michael and Mattias Werner. *The daVinci WWW Homepage*. URL: http://www.informatik.uni-bremen.de/ davinci/: University of Bremen, Germany, 1996.

33. Gleason, Howard Terrance. *Probabilistic Knowledge Base Validation*. MS thesis, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, 1995.

34. Gonzalez, Avelino J. and Douglas D. Dankel. *The Engineering of Knowledge-Based Systems*. Englewood Cliffs, NJ: Prentice Hall, 1993.

35. Harrington, Robert A., et al. "Development of an Intelligent User Interface for a Generic Expert System." *In M. Gasser (Ed.), Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference. URL http://www.cs.indiana.edu/event/maics96/Proceedings/harrington.html*. 1996.

36. Harrington, Robert A., et al. "GESIA: Uncertainty-Based Reasoning for a Generic Expert System Intelligent User Interface." *To appear in the Proceedings of the 1996 International Conference on AI Tools*. 1996.

37. Harrington, Robert A., et al. *The PESKI Intelligent User Interface*. Technical Report AFIT/EN/TR96-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1996.

38. Harrington, Robert A., et al. "Intelligent Interface Learning with Uncertainty." *Submitted to 1997 Florida Artificial Intelligence Research Symposium*. 1996.

39. Harris, Mary Dee. *Introduction to Natural Language Processing*. Reston, Viginia: Reston Publishing Company, 1985.

40. Hartrum, Thomas. *Class handouts, CSCE 594, Software Analysis and Design*. Technical Report, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1995.

41. Hartson, H. Rex and Philip D. Gray. "Temporal Aspects of Tasks in the User Action Notation," *Human Computer Interaction*, 7:1–45 (1992).

42. Hayes-Roth, Barbara. "An architecture for adaptive intelligent systems," *Artificial Intelligence*, *72*:329–365 (1995).

43. Heckerman, David. *A Tutorial on Learning With Bayesian Networks*. Technical Report TR MSR-TR-95-06, Microsoft Research, Advance Technology Division, Microsoft Corporation, 1995.

44. Hewitt, J. A. and P. G. R. Halford. "Design of an intelligent interface to standard PC applications which maximizes the ability of the disabled user," *Knowledge-Based Systems*, *6*:24–29 (1993).

45. Hook, Kristina. *Adapting to the User's Task*. Technical Report R95008, Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden, 1995.

46. Hook, Kristina, et al. "Inferring Complex Plans." *In W. D. Gray, W. E. Hefley, and D. Murray, editors, Proceedings of the International Workshop on Intelligent User Interfaces*. 1993.

47. Kanko, Mark. *Class handouts, CSCE 595, Software Systems Engineering*. Technical Report, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1996.

48. Kantorowitz, Eliezer and Oded Sudarsky. "The Adaptable User Interface," *Communications of the ACM*, *11*(32):1352–1358 (November 1989).

49. Karlgren, Jussi, et al. *The Glass Box User Model Filtering*. Technical Report R94014, Departments of Computer and Systems Sciences, Computational Linguistics, and Psychology, Stockholm University, Stockholm, Sweden, 1994.

50. Keller, Brian J. *A Practical Guide to X Window Programming*. Boca Raton, FL: CRC Press, 1990.

51. Kuhme, T. "User-centered approach to adaptive interfaces," *Knowledge-Based Systems*, *6*(4):239–248 (1993).

52. Kyburg, Jr., Henry E. "Uncertain Inferences and Uncertain Conclusions." *In the Proceedings of the Twelfth Conference for Uncertainty in Artificial Intelligence*. 1996.

53. Landay, James A. and Brad A. Myers. "Interactive Sketching the Early Stages of User Interface Design." *In Proceedings of CHI'95: Human Factors in Computing Systems*. 1995.

54. Landay, James A. and Brad A. Myers. "Sketching Storyboards to Illustrate Interface Behaviors." *In CHI'96 Conference Companion: Human Factors in Computing Systems*. 1996.

55. Lee, Geoff. *Object-Oriented GUI Application Development*. NJ: PTR Prentice Hall, 1993.

56. Luo, Ping, et al. "Management of Interface Design in HUMANOID." *In the Proceedings of the ACM Conference on Human Factors in Computing Systems, INTERCHI '93*. 1993.

57. Mason, Cindy L. "An Intelligent Assistant for Nuclear Test Ban Treaty Verification," *IEEE Expert: Intelligent Systems and Their Applications*, *10*(6):42–49 (1995).

58. Maulsby, David. *Inductive Task Modeling for User Interface Customization.* Technical Report Technical Report, Section on Medical Informatics Stanford University Stanford, CA, 1996.

59. Meyer, Beth. "Retail User Assistant: Evaluation of a User-Adapted Performance Support System." *In Lecture Notes in Computer Science, 4th International Conference, EWHCI'94*. 1994.

60. Miller, Christopher A. and Raymond Larson. "An Explanatory and Argumentative Interface for a Model-Based Diagnostic System." *Proceedings of UIST'92*. 1992.

61. Nakakoji, Kumiyo and Gerhard Fischer. "Intertwining knowledge delivery and elicitation: a process model for human-computer collaboration," *Knowledge-Based Systems*, *8*(2-3):94–104 (1995).

62. Neilson, Irene and John Lee. "Conversations with graphics: implications for the design of natural language/graphics interfaces," *International Journal of Human-Computer Studies*, *40*:509–541 (1994).

63. Ngo, L. and P. Haddawy. "Probabilistic Logic Programming and Bayesian Networks." *In the Proceedings of Asian Computing Science Conference, LNCS*. 1995.

64. Nicholson, A. E. and J. M. Brady. "Dynamic Belief Networks for Discrete Monitoring," *IEEE Transactions on Systems, Man, and Cybernetics*, *24*(11):1593–1610 (1994).

65. Nigay, Laurence and Joelle Coutaz. "A Generic Platform for Addressing the Multimodal Challenge." *In Human Factors in Computing System Proceedings, Annual Conference Series, SIGCHI'95*. 1995.

66. Nitsche-Ruhland, Doris. "A Knowledge-Based Authoring System for Hypermedia-Based Learning Environments." *In Lecture Notes in Computer Science, 4th International Conference, EWHCI'94*. 1994.

67. Oppermann, Reinhard. "Adaptively supported adaptivity," *International Journal of Human-Computer Studies*, *40*:455–472 (1994).

68. Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann, 1988.

69. Puerta, A. R. "The Study of Models of Intelligent Interfaces." *In the Proceedings of the 1993 International Workshop on Intelligent User Interfaces*. 1993.

70. Rook, Frederick W. and Michael L. Donnell. "Human Cognition and the Expert System Interface: Mental Models and Inference Explanations," *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(6):1649–1661 (November/December 1993).

71. Ruckert, Carsten and Stephen Klein. "Empirical Study on the Use of a Knowledge-based System for Selecting Standard Engineering Components." *In Lecture Notes in Computer Science, 4th International Conference, EWHCI'94*. 1994.

72. Rumbaugh, James and others. *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall, 1991.

73. Santhanam, Radhika and Susan Wiedenbeck. "Neither novice nor expert: the discretionary user of software," *International Journal of Man-Machine Studies*, *38*:201–229 (1993).

74. Santos, Jr., Eugene. *A Fully Integrated Probabilistic Framework for Expert System Development*. Technical Report, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1993.

75. Santos, Jr., Eugene and Darwyn O. Banks. *A Probabilistic Framework for Representing Uncertainty*. Technical Report AFIT/EN, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1995.

76. Schneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (Second Edition)*. Reading, MA: Addison-Wesley Publishing Company, 1992.

77. Stock, Oliviero. "Natural Language in Multimodal Human-Computer Interfaces," *IEEE Expert: Intelligent Systems and Their Applications*, *9*(2):40–44 (1994).

78. Stock, Oliviero and others. "AlFresco: Enjoying the Combination of Natural Language Processing and Hypermedia for Information Exploration." *In Mark T Maybury, editor, Intelligent Multimedia Interfaces, The MIT Press*. 1993.

79. Stock, Oliviero, et al. "Human-Computer Interaction through Natural Language and Hypermedia in AlFresco," *SIGCHI Bulletin*, *28*(3):102–107 (1996).

80. Sukaviriya, P. "From user interface design to the support of intelligent and adaptive interfaces: an overhaul of user interface software," *Knowledge-Based Systems*, *6*:220–229 (1993).

81. Terveen, Loren G. "Overview of human-computer collaboration," *Knowledge-Based Systems*, *8*(2-3):67–81 (1995).

82. Thomas, C. G. "Design, implementation and evaluation of an adaptive user interface," *Knowledge-Based Systems*, *6*:230–238 (1993).

83. Treu, Siegfried. *User Interface Evaluation: A Structured Approach.* New York, NY: Plenum Press, 1994.

84. Trumbly, James E. "Productivity gains via an adaptive user interface: an empirical analysis," *International Journal of Human-Computer Studies*, *40*:63–81 (1994).

85. van Zuylen, H. J. "From scientific computation to decision support," *Knowledge-Based Systems*, *6*(1) (1993).

86. Vaubel, Kent P. and Charles F. Gettys. "Inferring User Expertise for Adaptive Interfaces," *Human Computer Interaction*, *5*:95–117 (1990).

87. Waern, Annika. "Plan Inference for a Purpose." *In A. Kobsa and D. Litman, editors, Proceedings of the 4th International Conference on User Modeling.* 1995.

88. Watson, Mark. *Portable GUI Development with C++.* NY: McGraw-Hill, Inc, 1993.

89. Woods, D. D. "Price of flexibility in intelligent interfaces," *Knowledge-Based Systems*, *6*:189–196 (1993).

Appendix A. Basic Instructions for Application Access

A.1 Access to PESKI

The current version of PESKI is found at rharring/Sparks/PESKI_ALPHA on the AFIT Hawkeye network. Just type *peski20* at the command line to execute PESKI. PESKI help utilities includes explanation of all the interface buttons and choices, explanation of the basic PESKI functions, system programmer information, and tutorials for using some of PESKI's features. Help in using PESKI is obtained by choosing the Help menu choice of any PESKI menu. This action will start a Netscape [7] session to view the PESKI help in HTML. These HTML documents can be found offline from PESKI in the PESKI_Help_Files directory at rharring/Sparks/PESKI_ALPHA/ on the AFIT Hawkeye network. Any HTML viewer, such as Netscape, can be used to access the PESKI help information.

A.2 Access to daVinci

daVinci is a versatile graph drawing tool that provides an easy to use visualization environment. Information and downloading of the tool can be accomplished by accessing the daVinci World Wide Web (WWW) page [32]. A version of daVinci can be found at rharring/Sparks/PESKI_ALPHA on the AFIT Hawkeye network. Just enter the daVinci/daVinci_V2.0/ directory and type *daVinci* at the command line to execute the application. Also, some daVinci documentation is found at the same location in the /docs directory on the AFIT Hawkeye network in HTML format. Any HTML viewer, such as Netscape [7], can be used to access the daVinci help information.

A.3 Access to Netscape

Access to both PESKI and daVinci help utilities requires the use of an HTML viewer. The current viewer used for both application is Netscape [7]. Netscape can

be found on all AFIT networks, so see the appropriate network administrator for access information.

Appendix B. Physical Work Requirements

Symbol Key

mm = *mousemovement*

sc = *singlemouseclick*

dc = *doublemouseclick*

hk = *hotkeycommand*(< *cntrl* > *andaletter*)

1. Physical work requirements for startup choices without IIA - *Method 1*

   (a) Load BKB file = 4 mm, 2 sc, 2 dc

   (b) Choose the system function from the main window = 1 mm, 1sc

   (c) Choose the communication mode from the main window = 1 mm, 1sc

   (d) Execute the choice = 1 mm, 1 sc

   (e) Total work for *Method 1* = 7 mm, 5 sc, 2 dc

2. Physical work requirements for startup choices without IIA - *Method 2*

   (a) Load BKB file = 4 mm, 2 sc, 2 dc

   (b) Choose communication mode from main window menu = 1 mm, 2sc

   (c) Choose the system function from the main window = 1 mm, 2sc

   (d) Total work for *Method 2* = 6 mm, 6 sc, 2 dc

3. Physical work requirements for startup choices without IIA - *Method 3*

   (a) Load BKB file = 2 mm, 2 dc, 1hk

   (b) Choose the system function from the main window = 1 mm, 1sc

   (c) Choose communication mode from main window = 1 mm, 1sc

   (d) Execute the choice = 1 mm, 1 sc

   (e) Total work for *Method 3* = 5 mm, 3 sc, 2 dc, 1hk

4. Physical work requirements for startup choices without IIA - *Method 4*

   (a) Load BKB file = 2 mm, 2 dc, 1hk

   (b) Choose the communication mode from the main window menu = 1 mm, 2sc

   (c) Choose the system function from the main window = 1 mm, 2sc

   (d) Total work for *Method 4* = 4 mm, 4 sc, 2 dc, 1hk

5. Physical work requirements for startup choices with IIA - *Correct Suggestion*

   (a) Accept suggestion = 1 mm, 1sc

   (b) Total work = 1 mm, 1sc

6. Physical work requirements for startup choices with IIA - *Wrong Suggestion*

   (a) Reject suggestion = 1 mm, 1sc

   (b) Requirements from one of the four manual methods

7. Physical Work Totals

   (a) Total work with *Method 1* = 8 mm, 6 sc, 2 dc

   (b) Total work with *Method 2* = 7 mm, 7 sc, 2 dc

   (c) Total work with *Method 3* = 6 mm, 4 sc, 2 dc, 1hk

   (d) Total work with *Method 4* = 5 mm, 5 sc, 2 dc, 1hk

   (e) Total work with *Correct Suggestion* = 1 mm, 1 sc

   (f) Total work with *Wrong Suggestion* = varies

Appendix C. Survey Used for User Acceptance Testing

PESKI Graphical User Interface Evaluation Sheet

INSTRUCTIONS

1. Please follow the directions on this sheet. You will be answering questions in between steps so please, do not skip ahead at any time.
2. A test proctor will guide you through this process, but the proctor is not permitted to do or say anything that might alter your responses to the questions.
3. Remember, you are helping to judge the usability of PESKI's user interface, NOT THE FUNCTIONALITY OF THE SYSTEM. Try to concentrate on the interactive abilities of the user interface and answer the questions fairly.
4. When answering the questions, please be honest. Negative, positive, or neutral answers are all perfectly acceptable.
5. This survey should only take you about 10 minutes. Thanks for making an important contribution to my research!

Name:

Title:

E-Mail:

Please rate your expertise with using computer applications?

(Answer 1 thru 10; 1 is novice and 10 is expert)

QUESTIONS

1. Start PESKI. The command is peski20.
   (a) How would you rate the speed of the system's startup?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(b) List any miscellaneous comments you may have. If you don't have any, leave this question blank.

2. Delete the existing user logon called Dummy.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system when its deleting the logon?

   (Answer 1 thru 10; 1 is very slow and 10 is very fast)

   (c) List any miscellaneous comments you may have. If you don't have any, leave

3. Create a new user logon called NewUser. Set this user's class to be application user.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system when its creating the logon?

   (Answer 1 thru 10; 1 is very slow and 10 is very fast)

   (c) List any miscellaneous comments you may have. If you don't have any, leave

4. Login to the system with the logon NewUser.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system when its logging in?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(c) List any miscellaneous comments you may have. If you don't have any, leave

5. You will now receive two suggestions from the interface. Reject the suggestions.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system when rejecting the suggestions?

   (Answer 1 thru 10; 1 is very slow and 10 is very fast)

   (c) List any miscellaneous comments you may have. If you don't have any, leave

6. You will now receive two more suggestions from the user interface. Reject those suggestions as well.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system when rejecting the suggestions?

   (Answer 1 thru 10; 1 is very slow and 10 is very fast)

   (c) List any miscellaneous comments you may have. If you don't have any, leave

7. Load a BKB called rharring/Sparks/PESKI_ALPHA/BKBs/goldfish.bkb.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system to load the BKB?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(c) List any miscellaneous comments you may have. If you don't have any, leave

8. Close the BKB load window.

9. Start the Inferencing function in Structured Text communication mode.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system to start the inferencing task?

   (Answer 1 thru 10; 1 is very slow and 10 is very fast)

   (c) List any miscellaneous comments you may have. If you don't have any, leave

10. Close the Inferencing window.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) List any miscellaneous comments you may have. If you don't have any, leave

11. Exit PESKI.

   (a) How confusing is this task to perform?

   (Answer 1 thru 10; 1 is very confusing and 10 is very clear)

   (b) How would you rate the speed of the system to exit?

   (Answer 1 thru 10; 1 is very slow and 10 is very fast)

   (c) List any miscellaneous comments you may have. If you don't have any, leave

12. Start PESKI. The command is peski20.

   (a) How would you rate the speed of the system's startup?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(b) List any miscellaneous comments you may have. If you don't have any, leave

13. Login to the system with the logon NewUser.

(a) How confusing is this task to perform?

(Answer 1 thru 10; 1 is very confusing and 10 is very clear)

(b) How would you rate the speed of the system when its logging in?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(c) List any miscellaneous comments you may have. If you don't have any, leave

14. Accept suggestion to load ~harring/Sparks/PESKI_ALPHA/BKBs/goldfish.bkb.

(a) How confusing is this task to perform?

(Answer 1 thru 10; 1 is very confusing and 10 is very clear)

(b) How would you rate the speed of the system to load the BKB automatically?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(c) How beneficial is the intelligent assistant's suggestion?

(Answer 1 thru 10; 1 is useless and 10 is very useful)

(d) List any miscellaneous comments you may have. If you don't have any, leave

15. Accept suggestion to open Inference function in text mode.

(a) How confusing is this task to perform?

(Answer 1 thru 10; 1 is very confusing and 10 is very clear)

(b) How would you rate the speed of the system to start the Inferencing function

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(c) How beneficial is the intelligent assistant's suggestion?

(Answer 1 thru 10; 1 is useless and 10 is very useful)

(d) List any miscellaneous comments you may have. If you don't have any, leave

16. Close the Inferencing window.

(a) How confusing is this task to perform?

(Answer 1 thru 10; 1 is very confusing and 10 is very clear)

(b) List any miscellaneous comments you may have. If you don't have any, leave

17. Exit PESKI.

(a) How confusing is this task to perform?

(Answer 1 thru 10; 1 is very confusing and 10 is very clear)

(b) How would you rate the speed of the system to exit?

(Answer 1 thru 10; 1 is very slow and 10 is very fast)

(c) List any miscellaneous comments you may have. If you don't have any, leave

18. Please list any other comments you have.

Thanks Again for Your Help!

Appendix D. User Acceptance of the IIA

1. Average Computer Experience; scale is from 1 (novice) to 10 (expert)
   (a) Experience = 8.20
2. Average Opinion on Timeliness; scale is from 1 (very slow) to 10 (very fast)
   (a) Peski startup = 7.70
   (b) Login to system = 7.60
   (c) Loading an existing BKB manually = 6.80
   (d) Starting the inference window in structured text communication mode manually = 7.40
   (e) Loading an existing BKB through an IIA suggestion = 6.20
   (f) Starting the inference window in structured text communication mode through an IIA suggestion = 6.20
3. Average Opinion on Complexity; scale is from 1 (very confusing) to 10 (very clear)
   (a) Login to system = 7.00
   (b) Loading an existing BKB manually = 6.60
   (c) Starting the inference window in structured text communication mode manually = 8.60
   (d) Loading an existing BKB through an IIA suggestion = 8.00
   (e) Starting the inference window in structured text communication mode through an IIA suggestion = 6.60
4. Average User Opinion on Usefulness; scale is from 1 (useless) to 10 (very useful)

(a) Loading an existing BKB through an IIA suggestion = 7.20

(b) Starting the inference window in structured text communication mode through an IIA suggestion = 7.40

Appendix E. Responsiveness of the IIA

1. System startup from command line
   (a) Time using the IIA = 3.33 sec
   (b) Time without IIA = Fraction of a sec
2. User login
   (a) Time using the IIA = 6.89 sec
   (b) Time without IIA = Not Applicable
3. Start the inference engine window for text communication
   (a) Time using the IIA = 14.92 sec
   (b) Time without IIA = Fraction of a sec
4. Start the inference engine window for graphical communication
   (a) Time using the IIA = 18.50 sec
   (b) Time without IIA = 6.76 sec
5. Start the knowledge acquisition window for text communication
   (a) Time using the IIA = 14.83 sec
   (b) Time without IIA = Fraction of a sec
6. Start the knowledge acquisition window for graphical communication
   (a) Time using the IIA = 18.32 sec
   (b) Time without IIA = 6.73 sec
7. Start the edit supports window for text communication
   (a) Time using the IIA = 14.76 sec
   (b) Time without IIA = Fraction of a sec

8. Start the edit supports window for graphical communication

   (a) Time using the IIA = 18.34 sec

   (b) Time without IIA = 6.63 sec

9. Start the verification and validation window for text communication

   (a) Time using the IIA = 14.72 sec

   (b) Time without IIA = Fraction of a sec

10. Start the verification and validation window for

    (a) Time using the IIA = 18.45 sec

    (b) Time without IIA = 6.68 sec

11. Start the data mining window for text communication

    (a) Time using the IIA = 14.33 sec

    (b) Time without IIA = Fraction of a sec

12. Start the data mining window for graphical communication

    (a) Time using the IIA = 18.28 sec

    (b) Time without IIA = 6.75 sec

13. Start the knowledge viewing window for graphical communication

    (a) Time using the IIA = 12.04 sec

    (b) Time without IIA = 6.48 sec

14. Start load for an existing BKB file

    (a) Time using the IIA = 8.21 sec

    (b) Time without IIA = Fraction of a sec

15. Start load for a new BKB file (less than 5 existing BKB files)

(a) Time using the IIA = 10.48 sec

(b) Time without IIA = N/A

16. Start load for a new BKB file (greater than or equal to 5 existing BKB files)

(a) Time using the IIA = 13.33 sec

(b) Time without IIA = N/A

17. Reject both suggestions; load full5.bkb and start inferencing window in text communication mode

(a) Time = 40.38 sec

18. Accept both suggestions to load full5.bkb and start inferencing window in text communication mode

(a) Time = 31.25 sec

Vita

1st Lt Robert A. Harrington [redacted], although he calls Florida home. He graduated from Troy State University in 1993 with a Bachelors of Science in Computer Science. His 11 year career in the United States Air Force include Engineering Assistant, 834th Civil Engineering Squadron, Chief of Construction Management, 354th Tactical Fighter Wing(Operation Desert Shield and Storm), Satellite Software Test Analyst, 50th Space Systems Squadron, and Executive Officer, 50th Space Systems Squadron and 50th Logistics Support Squadron. Lt Harrington holds the John Levitow Award and is twice named Officer of the Quarter for the 50th Logistics Group.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2 Dec 96	Master's Thesis, 2 Dec 96

4. TITLE AND SUBTITLE
Utilizing Bayesian Techniques for User Interface Intelligence

5. FUNDING NUMBERS

6. AUTHOR(S)
1st Lt Robert A. Harrington

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr Abraham Waksman
AFOSR/NM
110 Duncan Ave
Bolling AFB, DC 20332 (202)-404-7496

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFOSR#94006

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this research is to study the injection of an intelligent agent into modern user interface technology. This agent is intended to manage the complex interactions between the software system and the user, thus making the complexities transparent to the user. The background study will show that while interesting and promising research exists in the domain of intelligent interface agents, very little research has been published that indicates true success in representing the uncertainty involved in predicting user intent.

The interface agent architecture presented in this thesis will offer one solution for solving the problem using a newly developed Bayesian-based agent called the Intelligent Interface Agent (IIA). The proof of concept of this architecture has been implemented in an actual expert system, and this thesis presents the results of the implementation. The conclusions of this thesis will show the viability of this new agent architecture, as well as promising future research in examination of cognitive models, development of an intelligent interface agent interaction language, expansion of meta-level interface learning, and refinement of the PESKI user interface.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Intelligent User Interface, Generic Expert System, Uncertainty-based Reasoning, Bayesian Network.	93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclass	Unclass	Unclass	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.